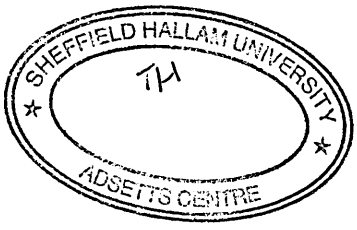
REFERENCE

Correlation of Localised Thermal Shock to Isothermal Biaxial Crack Growth Rates in Thin Plates of AISI 316 Stainless steel

Lee Edward Easterbrook (BEng. Hons.)

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University
for the degree of Doctor of Philosophy

August 2001


SHEFFIELD HALLAM UNIVERSITY
TH
ADSETTS CENTRE

Acknowledgements

I would like to express my thanks to Dr S. T. Hasan for his invaluable support and guidance during the work, and his support and advice during the preparation of this thesis. Without his guidance this work could not have been completed. I am also indebted to the technical staff of Sheffield Hallam University for providing me with the computing capacity and resources to complete the work. Specifically I would like to thank J. Stanley and S. Brandon of the technical support team for their tolerance and support and allowing my results files to eat their resources during my write up period.

Thanks are also due to the structural integrity group which provided stimulating discussion on a variety of subjects in a variety of places. Thanks must also be made to Prof. J. D. Atkinson in his role as head of research.

I am indebted to my friends and family for supporting me during this difficult time. To my colleagues of 4L12 1997-2000 I extend special thanks, without them, the three years would have been far less interesting.

L. E. Easterbrook

August 2001

Contents

Nomenclature

SYMBOLS

α	thermal expansion coefficient, K^{-1}, or angle of normal load to crack plane
ε	strain
θ	angle around crack tip, radians
Λ	ratio of lateral to normal load or stress
ρ	density, kg/m^3
σ	stress, Pa
$\Delta\sigma$	stress range
σ_{MAX}	maximum stress in a thermal shock cycle, Pa
σ_x	lateral stress component, Pa
σ_y	normal stress component, Pa
σ_{yield}	yield stress, Pa
σ_{xy}	shear stress component, Pa
σ_T	non-singular biaxial T-Stress, Pa
σ_L	lateral applied biaxial load, Pa
σ_N	normal applied biaxial load, Pa
σ_Q	lateral applied biaxial load to the crack tip, Pa
σ_P	normal applied biaxial load to the crack tip, Pa

NOTATION

a	crack length, mm
A, B, C	fitted constants
Bi	Biot Number
$B_{1,2,3}$	biaxial ratio function
C_P	specific heat capacity, J/kgK
FD	finite difference
FD&T	finite difference and Timoshenko
FE	finite element
Fo	non-dimensional time or Fourier Number
h	heat transfer coefficient, W/m^2K
k	thermal conductivity, W/mK
K_I	mode I stress intensity range, $Pa.m^{-3/2}$
K_{II}	mode II stress intensity range, $Pa.m^{-3/2}$
L	length

Q_f	heat flow by convection, J
Q_k	heat flow by conduction, J
r	radial distance from crack tip, mm
r_P	plastic zone size, mm
R	R-ratio, ratio of minimum load to peak load of $\Delta\sigma$ in isothermal loading
t_P	time period of thermal shock cycle, s
T	temperature, °C
T_A	Ambient temperature or cooling medium temperature, °C
T_{END}	temperature at shocked surface at the end of a thermal shock cycle, °C
T_{FACTOR}	thermal gradient function; fraction of, T_{END}, $T_{INITIAL}$ and T_A
$T_{INITIAL}$	Initial conditions of thermal shock, °C
T_P	nodal temperature, °C
T_N	northward nodal temperature, °C
T_S	southward nodal temperature, °C
T_E	eastward nodal temperature, °C
T_W	westward nodal temperature, °C
T_P^{t+1}	temperature, T_P one time increment in the future
Δt	time increment
Δy	element length, m

List of Figures

List of Tables

Abstract

The work presented covers numerical analyses of edge-cracked thin plates of austenitic stainless steel, AISI 316, loaded with transient thermal downshock. Further to this isothermal centre-cracked plates loaded under isothermal biaxial conditions are modelled. Crack growth rates are then applied to the models to determine a correlation between the two geometries.

The work presented demonstrates that a correlation exists between thermal downshock and biaxial isothermal crack growth rates. For such a correlation to be determined, the thermal shock and isothermal biaxial loading must be studied with an elastic-plastic material response. It is also determined that biaxiality in thermal shock is a function of the shock localisation due to localised contraction of the specimen in the direction of heat flow, making it necessary to evaluate thermal shock stresses in a two-dimensional sense. The relationship between biaxiality and localisation is found to be non-linear. At shocked regions greater than 25% of the available area maximum stresses remain largely unchanged. Below 25% of the available area maximum stresses drop significantly, by up to 50%.

Finally a method is proposed whereby thermal shock crack growth rates can be estimated by the bounding conditions of isothermal biaxial loading. This method uses an estimation of the heat transfer coefficient determined through a thorough analysis of a broad range of thermal shock cycles for AISI 316. Using the defining temperatures and time period of the cycle the h value can be estimated to within 3%. Once known the h value is fed into a simple log function describing maximum thermal stress, which can then be converted for any localisation present. It is also found that the correlating isothermal peak load is approximately equal to that of the thermal stress calculated. From this a modified Paris law is proposed to predict two sets of crack growth rates; one at equibiaxial conditions and the second at a biaxiality determined from the thermal shock load. This is shown to be a minimum of 0.35 at maximum localisation. This will calculate the upper, and lower, bounding limits of the thermal shock crack growth rates, providing a good estimation of the thermal shock crack growth rates.

1 – Introduction

Thermal shock is a severe form of thermal loading. Much work is conducted today to study its effect on reactor pressure vessels due to the accident scenario known as the 'loss of cooling event'. It can also occur on turbine blades and gun barrels in the form of rapid heating. Events creating thermal shock therefore, occur primarily in hazardous environments making understanding its nature on vital components such as reactor pressure vessels one of great importance. As a result of this severe form of loading, the high temperatures required and the rapid extraction or input of heat makes the experimental study of thermal shock expensive.

The subject of this thesis is to determine a correlation between the severe forms of loading in thermal shock fatigue and its correlation to bounding biaxial isothermal crack growth rates. To determine such a correlation would allow a much cheaper evaluation on estimating the severity of thermal shock loading. To this end numerical models have been conducted on transient heat transfer, linear-elastic and elastic plastic stress analyses and fracture mechanics. These analyses have been complemented with experimental data determined from a previous study by S. T. Hasan. A review of the work conducted in thermal shock and isothermal biaxiality is made and the contributions of numerical analyses noted. A successful method of analysis is then proposed.

Methods of the various analyses conducted are described in chapter 5 and their results presented in chapter 6. A discussion is then made in relation to the correlation of thermal shock to isothermal biaxial crack growth rates in chapter 7 and conclusions drawn and recommendations for further study made in chapters 8 and 9.

Chapter 5 is divided into 4 separate sections. Both theoretical and analytical analysis is covered, for thermal shock temperature and stress distributions, and determination of crack tip descriptors. This includes the mode I stress intensity range and a new plastic zone parameter derived for the purpose of this work. The parameter determines the biaxial form of the plastic zone coupled with its magnitude to perform a complete description of the plastic zone under linear-elastic and elastic-plastic material responses. Presented with this parameter is its theoretical determination from linear-elastic fracture mechanics for a von Mises mixed mode (I, II) plastic zone

including the non-singular term of biaxiality. Also presented is a method by which explicit definitions of the thermal shock cycle can be used to determined the state of loading and the proposed form such functions must take.

Chapter 6 covers the results of theoretical, finite difference and finite element models showing the important effects of localisation and elastic-plastic material response under two-dimensional analyses. The end functions describing the state of thermal shock loading and their relationship to isothermal biaxial models are also presented using the defining thermal parameters of thermal shock, i.e. initial conditions, end temperature of the shocked surface, ambient temperature and time period.

2 - Literature Review

2.1 – Overview of Fatigue & Fracture

There are many kinds of fatigue failure attributed to the micromechanisms at work in the material and the type of load cycle applied. However, the general pattern of crack propagation is the same and shown by Figure 1. Three stages are identified, Stage I - initiation, Stage II - propagation, Stage III - unstable and fast fracture.

Stage I is open to interpretation, just when a crack becomes a crack can depend on the perspective of the designer. Probably the best description is that of threshold - defined as a minimum range of a crack tip descriptor below which no crack propagation will occur, it is consequently a function of both loading and crack length. This makes the definition of minimum crack length only half of the problem, certainly when crack growth rates are of interest . However, it is agreed that initiation of a crack is generated at the microscopic scale by slip band deformation. Figure 2 is a schematic of the effect. Continuous cycling can create build-ups of these slip bands either as extrusions or intrusions effectively creating a stress raiser at the microscopic scale. Cracks have been found to initiate at the root of these slip-band positions [1]. Towards the end of the stage I region a threshold value, ΔK_{TH}, is found below which no propagation occurs.

A model of stage II crack propagation is presented by Tomkins [2]. This region is the primary region of growth for fatigue cracks and Tomkins' model demonstrates a repetitive deformation of crack tips through applied cyclic loads. The model is illustrated in Figure 3 where a fully compressed crack, Figure 3a, is loaded through a tensile/compressive cycle. On increasing the load through the tensile cycle the crack opens and 45° shear bands create plastic flow ahead of the crack tip, Figure 3b. At the tensile strain limit, Figure 3c, shear decohesion of the material occurs by these bands creating new crack surface. On the compressive reversal of loading the crack tip closes without recohesion of the new crack surface, Figure 3d. On returning to full compression a deformation of new surface occurs in the form of a ripple. Under repeated cyclic loading these ripples repeat with each cycle creating the familiar fatigue crack surface effects of striations. Quantification of this region of crack propagation was made by Paris and is known as the Paris Law;

$$\frac{da}{dN} = C(\Delta K)^m$$

eqn. 1

The Paris law provides a logarithmic relationship between the crack growth rate, da/dN, and a crack tip descriptor, K - the stress intensity factor.

The stress intensity function, K, was first quantified by Irwin in 1957 [3] as a simple function of the force tending to cause crack extension and today is generally given as a function of applied stress and crack length;

$$K = \sigma\sqrt{\pi a}$$

eqn. 2

Erdogan gives a full description of K and further detailed analysis [4]. However, the function must reflect the conditions and geometric constraints to which the crack tip stress field is exposed. This is usually achieved through a Green's function method with simplifications taken against geometry and loading. Even then such functions can be extremely difficult to use in terms of the integral equation resulting from the Paris law. Much work has been conducted to this effect and presented in data books, [5]. A descriptive form is,

$$K = \sigma\sqrt{\pi a}\ldots\left(\frac{a}{W}\right),\left(\frac{W}{a}\right)$$

eqn. 3

where a is the length of an edge crack or half the length of a contained crack and W is the overall length of the cracked plane.

Further description of the intensity of loading is provided by ΔK - the range of stress intensity factor between the maximum and minimum tensile values of K as determined by the stress cycle. The ratio of these values, R:

$$R = \frac{K_{MIN}}{K_{MAX}} = \frac{\sigma_{MIN}}{\sigma_{MAX}}$$

eqn. 4

is found to increase crack growth rates as it increases, behaviour more prominent in stages I and III.

The constants C and m are required parameters to carry out the analysis. They are both related to material and loading conditions and m is commonly said to vary

between 2 and 4. Since a logarithmic scale is adopted, m describes the gradient of stage II.

Stage III, introduces a critical point at which the crack will accelerate to catastrophic failure. This is described by a critical stress intensity range, ΔK_C, which is found to decrease substantially with increasing thickness of specimen. As plane strain conditions approach the value falls to a constant. Thick specimens for plane strain conditions are therefore needed in order for such a critical stress intensity to be used as a meaningful measurement of material fracture toughness.

The fracture toughness of a material can be a consequence of very different micromechanisms at work within the material. Original work into the fracture of materials was carried out by Griffiths in the 1920's on brittle materials - primarily glass with elliptical cracks. Today fracture is grouped into either brittle or ductile fracture and Figure 4 shows some of the main mechanisms involved. To determine the type of fracture (brittle or ductile) in a crystalline material means identifying whether or not plastic flow is required for material separation. From a brittle standpoint fracture primarily occurs from the separation of atomic planes, while requiring high stress to separate planes the energy involved to create the new surfaces is small [6]. Ductile fracture, however, requires around 10^6 times more energy to separate material since plastic flow has to take place.

The primary mechanism of ductile fracture is due to nucleation of microscopic voids, either at dislocation pile ups, points of severe deformation, or around the interface of particle inclusions. These voids grow in size ahead of the crack tip where the stress/deformation field is greatest, increasing in concentration until they coalesce to create larger voids or separate completely to form further crack surface. As with brittle fracture, ductile fracture can occur by transgranular or intergranular fracture illustrated by Figure 4. This is dependent on the amount of slip planes present in the material structure; the more slip planes present the more dislocation pile ups occur at grain boundaries and therefore the greater energy present to initiate void growth and crack propagation. Consequently, face centred cubic (FCC) materials, having more slip planes than body centred cubic, are more likely to fracture by intergranular processes [7,8]. At high temperatures creep effects can also be present. These processes can combine to generate complex fracture processes, to study this, further

work has been presented by Miller [9] and Ashby [10] to group the mechanisms at work in different materials as conditions of temperature and loading.

Both ferritic and austenitic steels demonstrate a range of fracture processes. Austenitic steels specifically can fracture under low loading and long times by intergranular processes. However, at high temperature transgranular creep fracture dominates. This can, however, be returned to intergranular fracture if inclusions or precipitates are present at the grain boundaries, fracture will occur due to their effect as a stress raiser.

For AISI 316 the mechanisms of fracture are difficult to isolate. High levels of dislocation pile ups at grain boundaries, generating high strain energies, induce the precipitation of carbide particles at grain boundaries. This occurs primarily at temperatures ranging from 500 to 700°C, which maintain the materials stiffness at these temperatures. A general trend of transgranular fracture is found in thermal shock (TS) tests by Marsh [11,12] where oxide penetration is also noted at all crack initiation sites. Whereas Miller presents data on monotonic loading for creep and shows for the above temperature region, with yield order loads, intergranular creep fracture dominates.

It is apparent for a full description of fracture the immediate conditions taking place must be known and quantified. Fatigue and fracture spans into many different fields which can be divided into two categories, environmental fatigue/corrosion and mechanical fatigue/fracture. The latter being the focus of the current work, with its various branches shown by Figure 5. Essentially groups are defined in terms of mechanical and temperature types of loading. The groups have evolved with attempts at furthering the accuracy of predictive methods and together describe the environments engineering components are exposed to.

2.1.1 – Fracture mechanics

Classical linear-elastic fracture mechanics (LEFM) has long been established. Over past decades it has been the subject of much work resulting in the fields divergence to many varied areas of engineering. As a result a few of the key papers and those relevant to this work will be mentioned. Tomkins [2] furthered the field by presenting a thorough analysis to quantitatively assess the mechanism of fatigue crack

growth in metals. Centred on the propagation of a crack by damage in the shear planes the basic theoretical laws governing fatigue failure at both high and low stress regions were derived, with material parameters controlling crack growth presented. Made in conjunction with experimental work the theoretical and experimental findings compared favourably throughout the fatigue regime and confirmed low stress crack propagation previously presented by Paris.

LEFM is however limited to low loads and stress fields in relation to a materials yield stress and the field of fracture mechanics has always searched for a method of equal rigour to be applied under higher loading conditions in the elastic-plastic (EP) region. To this end parameters such as crack opening displacement (COD), crack tip opening displacement (CTOD) and plastic zone (r_P) have been looked at. However, an elastic-plastic fracture mechanics (EPFM) analysis does not generally present a complete solution.

In 1968 [13], furthered in 1976 [14] and 1988 [15] Rice presented a parameter known as *J*, the path independent contour integral. The parameter was based on the accumulated strain energy around a contour taken from one crack face to the other. It was unique in that the value of J was independent of the path chosen. This made the parameter in later years ideal to finite element analyses. Based on strain energy the parameter was not subject to the limits of linear-elastic (LE) models and could be equally applied to both LE and EP material responses. Mathematically equivalent to the energy release rate (G), the J-Integral could be related back to COD and K values under linear-elastic conditions and approximate estimates to these were made. Rice also showed the integrals capability to deal well with blunt crack tips under both LE and EP conditions.

The J-Integral was also expanded to use in mixed mode fracture mechanics, an excellent derivation of which is presented by Hellen [16]. An illustration of this derivation was put well by Chen [17] in 1981. Here the determination of J is orientated between two different orthogonal limits resulting in two separate values of J referred to as J_I and J_{II}. It is important to note that these are simply values of J and do not represent equivalent modal stress intensity values. J_I and J_{II} can also be described as functions of K_I and K_{II} and thus can be determined by solving simultaneously. This renders the J-Integral even more useful to the methods of finite element.

In 1988 Lambert *et al* [18] evaluated the AISI 316 thick plates for crack propagation using J as a fracture parameter. Using both numerical and theoretical

methods of calculating J a conservative Paris correlation was found and proposed as a calculation method for generalised plasticity in structures. Banks-Sills *et al* [19] in 1991 used an empirical relationship to calculate ΔJ by integrating the load vs. displacement curve and comparing to J values determined through G equivalence. Again the ΔJ parameter was successfully used in a Paris law correlation, Mao *et al* [20] presented the use of ΔJ in the damage tolerance of a compressor blade. Placed under centrifugal and gas forces a critical J was already known and the corresponding critical crack length was determined.

As well as the use of J and COD the plastic zone, r_P, presents itself as a promising indicator to the state of a crack tip. Since r_P is a direct result of the crack tip it must therefore be an ideal indicator to the prediction of crack propagation. The plastic zone is long established as a fundamental attribute to LEFM, however, it has never been able to demonstrate itself as a thorough crack propagation parameter. Due to its direct relationship with crack tip stress the plastic zone can become more descriptive of specimen geometry and general yielding under high load scenarios than the singular stress about a crack tip. It is tempting, however, as a biaxial parameter since the non-singular term describing T-Stress can be introduced to its description. Consequently the r_P could be used as a measure determined by numerical methods for understanding the state of loading about the crack tip. On the other hand it could also be used under known mixed mode loading as a predictor to the direction and rate of crack propagation. Some recent work has used the r_P to determine the effects of biaxiality in various conditions.

Bobet *et al* [21], used the r_P in conjunction with critical direct and shear strengths of gypsum rock under uniaxial and biaxial loading. The r_P was used in numerical models to determine the onset of initiation in brittle fracture, showing a rigorous application to results from external works. Bhargava *et al* [22] also used r_P with the Dugdale model to describe biaxially loaded cracks. Shaniavski *et al* [23] also used r_P in modelling crack growth in aluminium alloys. The plastic zone was used under various R-ratios and biaxial loading scenarios. However the orientation of the plastic zone was not used to determine the direction of crack propagation. It was noted however, the plastic zone was not linearly related to the ratio of biaxial loading. This is analytically determined in the work presented within this thesis, where limits of numerical stability are determined. Plastic zone is not consigned to numerical

analyses, it has been studied via transmission electron microscopy by Farkhutdinov *et al* [24] in 1995 to study the plastic zone under impact bending tests after superplastic thermomechanical treatment. It would seem therefore that r_P is an equal contemporary candidate for a crack tip descriptor as the J-Integral mixed mode K values. The capability of r_P, however, to maintain a realistic description when it is large, compared to the crack length is still questionable. A possible by-product of this is the use of plastic zone shape around notches containing cracks in elastic-plastic models to determine crack growth as conducted by Ahmad *et al* [25].

2.1.2 – Isothermal Biaxiality Fatigue

Isothermal fatigue is the simplest form of cyclic fatigue. The majority of engineering components can be described as operating at a constant and uniform temperature. Along with the cyclic loads of service this temperature is the only external parameter of interest and is used only in selection of appropriate cyclic/mechanical material properties. Indeed other more complicated forms of cyclic loading are sometimes referred back to a single representative isothermal temperature. Until recently isothermal fatigue tests have been conducted on simple plane strain uniaxial compact-tension (CT) specimens. In practice however, a component is usually subjected to a more complex stress system than uniaxial. Biaxial stress fields, for instance, are set up by structural constraints, changes in geometry, ductile behaviour or simply loaded in a non-uniaxial fashion during service. Therefore in today's more demanding economic climate and high competition for accurate component specification in life as well as in performance more complex methods of evaluation are required.

The Paris law has been successfully used for uniaxial fatigue test analysis. For a similar method to be determined for biaxiality then the dependency of crack growth rate on ΔK as derived from surrounding biaxial conditions must be known. Since the Paris law predicts crack growth rates that are only a function of stresses affecting the crack tip stress field the consequence of biaxiality and the possibility of mixed mode crack growth must be evaluated.

In 1974 Miller and Kfouri [26] compared the performance of LE to EP models using various crack tip parameters including J-integral and plastic zone size under

biaxial loading. It was concluded that for LE materials the stress intensity factor adequately defines the crack tip state, whereas in EP materials more complex descriptions are required. This is contrary to work later submitted by Eftis *et al* [27,28] who described the requirement of an additional non-singular term in the descriptions of stress and displacement around a crack tip when dealing with any form of biaxiality. In the same year Kfouri and Miller demonstrated the changes in fracture toughness parameters J and G^{Δ} under biaxial conditions [29]. In the following year, Kfouri and Miller [30] demonstrated that J and plastic zone can be used to described the biaxial stress state about a crack tip.

Brown and Miller [31] have designed a biaxial thin-plate test-specimen for plane stress scenarios. This 'cruciform' specimen is able to carry out both biaxial and uniaxial loading tests, whilst still maintaining a biaxial effect. The design is such that in a uniaxial test the free edges are constrained from displacing inwards so creating an effective biaxial system. A similar specimen of more complex geometry, and designed for both horizontal and slant cracks has been used for biaxial tests on weldable metals by Kitigama *et al* [32]. Charvat *et al* [33] designed a test rig for the study of crack growth rates of biaxial loading systems. Using similar specimens to Charvat, Rhodes and Radon [34] looked at the inherent local biaxiality ahead of a crack tip in aluminium alloys finding applied biaxial loads manipulated the local biaxiality. Kfouri and Miller [35] have also investigated inclined cracks in biaxial stress fields using finite element models of Brown and Miller's cruciform specimen.

Initial definitions of the applied and resulting biaxial stresses are described by two parameters, σ_T and Λ, where;

$$\sigma_T = \sigma_L - \sigma_N$$

eqn. 5

$$\Lambda = \frac{\sigma_Q}{\sigma_P}$$

eqn. 6

where σ_L and σ_N are the lateral and normal applied stresses respectively, σ_P and σ_Q are the principal stresses at the crack tip. For non-rotated cracks, σ_T would be fully described with σ_L and σ_N by;

$$\sigma_T = \sigma_N(\Lambda - 1)$$

$$\Lambda = \frac{\sigma_L}{\sigma_N}$$

eqn. 7

Under increasingly negative values, i.e. increasing normal stress (a process of Mode I crack growth) its effect is thought to accelerate mode I cracks along with increasing plastic zone size and crack opening displacement (COD), with $\Delta\sigma_N$ governing the magnitude of the effect. Interestingly, Kfouri also found a reduction in crack tip stress whilst supporting the use of σ_T to describe biaxiality along with mode I and II stress intensity factors. Furthermore, rotated cracks are found to adjust themselves to a path where $\Delta K_{II} = 0$ or to a path corresponding to Mode I loading.

All previous biaxial tests have been presented in terms of the Paris equation. However, no single unambiguous method of representing the crack tip state under biaxial loading has been made, i.e. expressing K as a function of Λ. Due to what appears to be a dominance of mode I loading on a crack tip under biaxiality such enhanced consideration may not even be necessary [31].

At high temperatures (≈0.4+ of melting temperature) creep fatigue takes place. A material of previously mentioned studies by Brown and Miller in this thermal region is that of AISI 316. In 316 creep fatigue occurs at around 600-700°. At these temperatures oxidation of new crack surfaces can take place - a mechanism associated with crack closure and thought to ease the effective stress intensity factor.

A thorough review of the early biaxial work carried out is made by Smith and Pascoe in 1983 [36]. The review found contradicting results, where biaxiality was said to accelerate, decelerate and affect in no way at all the growth of fatigue cracks. Several suggestions were put forward as reasons for this, including geometry, testing parameters, material behaviour and the use of Paris law reducing the ability of small changes to be observed. Therefore the effect of biaxiality is not absolute, like all other aspects of engineering it is dependent on different factors. It can be said however, that biaxiality is certainly manifest in theory and in numerical practice at the level of the crack tip singularity and should therefore be of direct relevance to the quantification of complex loading.

2.1.3 – Thermal Fatigue

Thermal fatigue can be known by many different names - creep fatigue, thermal shock, craze cracking, thermal rupture, thermal strain fatigue, thermal stress fatigue and high temperature fatigue. All these names isolate the importance of one effect on fatigue performance - temperature. Spera [37] has defined thermal fatigue as 'the gradual deterioration and eventual cracking of a material by alternate heating and cooling during which free thermal expansion is partially or completely constrained'. In other words basic fatigue with loads induced from thermal expansion/contraction as opposed to externally applied loads. An exception is creep fatigue, which can not solely be described by thermal fatigue; since creep effects are observable at high constant temperatures (isothermal fatigue) as well as high cyclic temperatures (thermal fatigue). Creep-fatigue interactions are an important mechanism and their relevance to a situation must be evaluated in high temperature models. However, AISI 316 is sufficiently unique in its high temperature cyclic behaviour to allow a more in depth approach to creep models. Rios *et al* [38] shows that AISI 316 generates dislocations at temperatures above 550°C creating non-uniform patterns in material behaviour with increasing temperature. By way of short summary work has been conducted [39,40,41,42,43,] on creep behaviour of AISI 316 and found to emphasise above 550°C the importance of oxidation and precipitation of carbides on fatigue life. The effects of mechanical dwell periods are also reported to drastically reduce the fatigue of AISI 316.

Quantifying a crack tip under thermal loading conditions is not as easily made or readily available in literature than that of isothermal mechanical loading, though the first major study towards this was the work conducted by Coffin [44, 45] where the effects of thermal cycling on ductile metals was investigated. It has also been confirmed by Sih [46] that Irwin's concepts of brittle fracture can be applied to determine stress intensities from thermoelastic stresses. This led to later work conducted by Emery [47] to determine the analytical solution of an edge-cracked plate subject to thermal stresses.

Figure 6 shows a long and thin specimen containing a small edge crack perpendicular to its length, either fully or partially constrained. Under such simplistic geometry a thermal cycle could be attributed to mode I loading, since thermal expansion or contraction would generate compressive or tensile stresses respectively.

This would infer standard use of the stress intensity function K in terms of the effective mode I stress from constraints of thermal expansion. For the elastic regime simple thermal stresses can be found by;

$$\sigma = \alpha E \Delta T$$

eqn. 8

where E = Young's Modulus, α = coefficient of thermal expansion, ΔT = change in temperature. Taking into account R-ratios a stress range would be written as;

$$\Delta\sigma = \alpha E \Delta T (1 - R)$$

eqn. 9

However, specimens will rarely lend themselves to such simple geometries and the multiaxial effects of thermal expansion may have to be accounted for. As such a singular interpretation of mode I, II or III loading is not sufficient, crack tip descriptions using K must be approached in specific terms of temperature and thermal stresses. In isothermal analyses K is defined from derivatives of stress functions in two dimensions which in turn are taken from the considerations of strain. Temperature introduces another dimension, by further complicating analyses to such a degree that a single value of temperature is inevitably taken as representative of the thermal cycle. Simplifications may even be made to calculate the primary stresses induced by thermal cycling and used as a basic mechanical approach. As mentioned in the previous section biaxial loading systems could be used to attempt to model transverse and in-plane loading of thermal expansion. This may lead to a dominance of model I loading losing accuracy of any biaxial thermal stress system. Adopting a purely mechanical approach leads to the fundamental lack of modelling temperature dependant properties of the material. As with using a single temperature to describe a thermal cycle, single values of material properties may be used in the analysis. Though the larger the temperature range of the cycle the more gross the simplification. This is clearly shown by Kokini [48] who showed when significant changes in thermal properties occur, including, specific heat, thermal conductivity and expansion, drastic under approximations can be made in stress intensity factors.

It is here that computer power can be harnessed by using finite element methods. Crack descriptors such as the energy release rate (G), J-Integral (J), crack opening displacement (COD) and crack tip opening displacement (CTOD) can be calculated relatively easily from nodal displacements of a mesh model. Hence any form of

loading can be made in a model along with temperature dependant material properties and the required information gained by sub-routines from the model results. The commercial package ABAQUS is able to calculate stress intensity values for surface cracks whose depth is capable of increasing on a small scale. Any of these values determine a description for a static crack.

The majority of service loads involving thermal fatigue are substantially less complicated than mechanical loading situations, since thermal fluctuations are generally the result of machine start-up and shut-down. Elevated or even sub-zero temperatures are often held at an approximate constant where start-up and shut-down of the system invariably causes most damage. Figure 7 shows basic representations of the thermal load cycle. Turbine blades in a gas generator are prone to the kind shown by Figure 7a where rise and falls in demand require the start-up and shut-down of further generators. A similar form would be that of turbine blades in a jet engine shown by Figure 7b where take-off and landing expose the blades to large increases in temperature whilst flight time incurs a steady-state and overall lower temperature.

The steady-state periods are the equivalent of mechanical dwells in isothermal fatigue cycles. It is generally agreed that such mechanical dwells at high temperatures are sufficiently damaging. However, thermal dwells need to be well within the creep region to create such damage.

It is rare however, for thermal fatigue to be considered exclusive of any form of mechanical loading. Components tend to be constrained in operation or sufficiently loaded to require application of mechanical loads as well. More detailed high temperature operating environments can be accounted for if mechanical load cycles are included and superimposed with thermal load cycles to create another area termed thermomechanical fatigue. An area discussed in the following section.

2.1.4 – Thermomechanical Fatigue

Thermomechanical cycling is used to model situations which cannot be accurately tested by straightforward isothermal tests with uniaxial data. Thermal and mechanical load cycles are applied in two ways, out-of-phase (OP) and in-phase (IP) shown in Figure 8. Loading scenarios where the maximum mechanical strain applied coincides with the maximum temperature are termed in-phase and similarly where the

minimum applied mechanical strain coincides with the maximum temperature are out-of-phase scenarios.

In order to discover if simpler isothermal tests are a viable alternative comparisons were made by Miller and Priest to those of isothermal tests using the maximum temperature of the thermal cycle with varying degrees of success [49]. IP cycles are considered to be the most damaging since maximum thermal strains are superimposed on maximum mechanical strains, essentially a two fold effect, whilst OP cycles are a contrary scenario such that any maximum mechanical load will not be supported by a low thermal strain. These conditions, however, are dependant on experimental conditions. If, for instance, an R-ratio of -1 is used for both cycles then thermal contraction will be generated at the same instant of mechanical tensile straining and a maximum 'cancelling' effect will take place. If an R-ratio 0.1 is used then the effect will be similar but not as pronounced.

Work conducted by Beauchamp [50] on AISI 316 in the range of 400-625°C shows that IP cycles are very accurately matched to those of isothermal tests whilst OP cycles provide excessive endurance. Beauchamp explained this not as an effect of net loading but in terms of a coincidental effect of fracture mechanisms.

In isothermal tests Beauchamp found fracture mechanisms to be intergranular nucleation followed by mixed mode crack growth. IP tests, however, revealed significant intergranular cracking whereas very little took place in OP tests. A lower mean temperature through thermal cycling would mean less surface cracking which would be compensated for by internal cracking in IP tests giving endurances coincidentally close to isothermal results. A lack of internal cracking in OP tests means that little overall damage takes place and so longer lives are a result.

Consequently, situations of thermomechanical fatigue require complicated models to verify any simplistic use of isothermal fatigue data.

2.1.5 – Thermal Shock Fatigue

It is fair to assume that a material has been sufficiently well chosen for its performance at elevated temperatures, hence cycles shown in Figure 7a and b become a matter of primary damage by large changes in temperature over a brief period of time. Indeed some situations are exposed only to this kind of cycle. Figure 7c shows

such an event by cooling systems in liquid metal fast breeder reactors using molten sodium as a coolant and heat extractor. By a combination of the sodium coolants high heat transfer coefficient and high conductivity, component surfaces exposed to the coolant follow thermal fluctuations very rapidly. This can occur during reactor shutdowns where power is quickly reduced. Marsh *et al* [51,52,53] has conducted work on such events with cooling rates of 10°C/s over 200°C initiating from 550-650°C. Other examples of similarly rapid temperature reductions given by Baron [54] are local quenching of piping by water droplets carried in superheated steam and gas turbine components when combustion is halted. From the opposite point of view Figure 7d shows the likely cycle for a gun barrel experiencing rapid increases in temperature during each firing and generating a distinct form of fracture known as craze-cracking. Northcott and Baron conducted work on rapid heating with high frequency induction and wedge shaped specimens showing this form of cracking [55].

This kind of cycle is described as a thermal shock cycle and is the most damaging form of low cycle thermal fatigue.

Thermal shock events primarily take place at elevated temperatures. As such the up-shock event (increasing temperature) is much more common as a repeating cycle than the down shock (decreasing temperature) event. This is due to an operations cooling rate commonly left to natural cooling by conduction and/or convection. However, as mentioned previously the less common downshock events do take place and in extreme environments.

One of the first accredited works into experimenting with thermal shock is that of Glenny *et al* [56], with evaluations of various methods for carrying out both thermal upshock and downshock tests within predetermined temperature ranges. Acknowledging the importance of thermal shock exposure on both brittle ceramics and metallic materials, methods presented are orientated around a disc specimen now known as the Glenny Disc. Figure 9 shows the specimen geometry used by Glenny *et al.* A tapered design is used to simulate wedge geometry of turbine blades incorporating the effects of a non-uniform section on the generation of thermal stresses. Methods including radiation and high frequency induction heating, forced convection and immersion in high temperature molten metals and salts were all rejected against each methods inability to maintain reliable values of heat transfer coefficients, damage to the specimen and control over temperature ranges produced.

The suggested method was that of fluidised bed heating or cooling of the specimen. A refractory powder is made fluid by passage of gas through a permeable support plate. Altering the rate of gas flow and conductive properties of both gas and powder high fluidity and high levels of heat transfer are obtained under controlled conditions. Rapid cooling can be achieved by plunging a hot specimen into a room temperature fluidised bed or rapid heating by raising the beds temperature with surrounding electrical resistance heaters. Bed temperatures of 1000°C were reported with higher temperatures achievable from a secondary heater.

Behaviour of brittle and ductile materials under thermal shock is similar to those of ordinary isothermal mechanical conditions. Brittle fracture tends to occur as a shattering of the specimen when multiple small cracks or craze cracking becomes unstable. Griffith's brittle fracture theory still holds - instability occurs when the elastic energy released in extending the crack exceeds the energy required to create new surfaces. Ductile behaviour can be accounted for by modifying the surface energy term to include the energy for local plasticity around the crack tip. This may prove to be somewhat more complicated for thermal shock since localised tensile and compressive stresses are set-up, as well as plastic deformation, not solely due to the crack tip.

The most common downshock event is that of quenching. It is well known that component warping can occur from residual stresses incurred from rapid cooling. Manson [57] studied the effects of single shock quenching and thermal cycling, producing parameters for material choice when quenching and thermal cycling are considerable loads. Skelton studied the effects of quenching on cylinders of austenitic steels [58] cycling the event to 30 000 cycles - finding austenitic steels accelerated cracks slower in the early stages than ferritic steels, though managing to penetrate further. Quenching is also a concern of the aerospace industry, where Ramakrishnan [59] using a hollow disc for a specimen, generated a designers information database to address solution heat treatment quenching. Similar analyses were also conducted by Askel [60] and Rasty *et al* [61] who determined the deflections of a beam under quenching by various numerical methods.

Probably the harshest example of thermal shock is that described by Seki [62] in 1991 for plasma facing components in the research programme of the international thermonuclear experimental reactor. Assessing different materials including type SUS

of 316 for containment of the plasma surround analysis of shocks of 1460°C over 60 milliseconds were attempted. Predictably the lack of material property performance at these temperatures hindered any meaningful results from the analysis.

Brown [63] and Hasan [64,65] have studied repeated thermal downshock cycles with additional mechanical loads. Using air jet cooling of hot specimens rapid cooling was obtained on edge cracked stainless steel plates. A strain intensity factor was used to correlate results to isothermal tests with some success. However, analytical derivation of fracture parameters was based on a one-dimensional system when it is possible a two or even three-dimensional thermal distribution was present. Correction of this discrepancy could result in a better correlation than was originally found. Rapid acceleration of crack growth at early stages was found which then fell into arrest. However, the application of a tensile uniaxial load of a similar order to the compressive mid region stress could maintain crack growth beyond the compressive region.

An introductory paper by Skelton [66] presents a good description of the stress-strain system induced by a thermal shock cycle. Figure 10a shows a schematic of applied up and downshocks on the edge of a thin plate while Figure 10b shows the resulting stress/strain hysteresis loop through the process of an up and down thermal shock cycle as illustrated by Skelton.

When the surface of a body is exposed to a medium of much lower or higher temperature, a large thermal gradient can be set-up through the slow response of thermal conductivity within the material. When an upshock is applied to a surface, its temperature will rapidly increase forcing surface into expansion. The bulk material, which could be up to 75% of the whole, still at the initial temperature will generate a resisting constraint against that expansion, forcing the surface into compression and creating a large negative stress gradient at the surface. With high temperatures and ductile material this will likely pass into plasticity to point *A* of Figure 10b. As thermal conductivity catches up, heat will spread through the remainder of the region initially unloading the material, reducing the stress and thermal gradient only to pass the whole into expansion to point *B* approximately at the original state of strain. Holding times at a constant temperature may relax the residual stress imparted. Application of a downshock will then reverse the process. The surface will attempt contraction but will again be resisted by bulk material throwing the surface into tension to point *C* until

thermal conductivity reduces the thermal gradient and hence the stress gradient. As the whole cools bulk material forces the yielded surface into compression, point *D*. After repeated cycles the system reduces to a closed hysteresis loop initiating and propagating fatigue cracks by reversed plasticity. Such a dual event Skelton attributes to the start-up and shut-down of power plants.

Since under downshock situations a surface is instantly thrown into tension generating plastic deformation in the very region where a crack will initiate it is the most damaging of the two. As such it is downshock models that will be the focus of this research.

2.1.6 – Pressurised Thermal Shock

A review of thermal shock would not be complete without a description of the work undertaken in recent years on pressurised thermal shock (PTS). Created by use of the emergency core cooling system of a reactor pressure vessel (RPV) thermal shock can take place under pressurised conditions increasing cooling rates and introducing possible mechanical loads. Sievers *et al* [67] obtained a simplified method to determine the stresses and stress intensity factors through the pressure vessel wall under this form of loading. Sievers concluded that a particularly complex loading system was present making triaxiality an important factor in correlating results. This is supported by Chawla [68]. The importance of PTS in the modern power generation industry is clear when it is the subject of significant safety work on specific reactors, [69,70]. Further to this the available fracture mechanics concepts have been evaluated by Roos *et al* as late as 2000 [71] to ensure they adequately represent the fracture of an RPV.

Much work has been done in recent years to develop the numerical simulation of damage and crack propagation of RPV's subject to PTS. Probabilistic fracture mechanics (PFM) has been used [72,73], where various required inputs to the model, such as geometry, material properties, crack length and shape etc. for RPV's are altered to evaluate the designs sensitivity. This allows the consideration for the origin of most likely failure to be quantified. Numerical simulations into the propagation of cracks is conducted by Zhao *et al* [74], where the possibility of continued crack propagation as opposed to arrest is studied in terms of crack tip channelling and

spalling. The effect of crack orientation under PTS is studied by Jang *et al* [75], determining that screening methods did not represent the same level of risk to circumferential flaws than axial ones. Matsubura [76] looked at the effect of the heat transfer coefficient on the relationship between crack tip parameter J and non-dimensional crack length, concluding the shape of the relationship is governed more by the material constants of a Ramberg-Osgood type model than the heat transfer coefficient.

Finally in 1999 Bass *et al* [77] designed a cruciform specimen to study the effects of PTS on surface flaws under uniaxial and biaxial conditions. It was successfully demonstrated that equibiaxiality could reduce fracture toughness by 42% at certain temperatures, whilst at lower temperatures it appeared to have no effect. This supports the previous review by Smith and Pascoe that material behaviour is a key factor in the response to biaxiality. It was also shown that principal stress analysis did not illustrate these effects.

2.2 – Numerical Analysis in Continuum & Fracture Mechanics

2.2.1 – Continuum Mechanics

Continuum mechanics is the application of strength of materials theory to a component model in order to determine its performance under certain constraints and loading conditions. For linear elasticity the equilibrium and compatibility equations are satisfied by Airy's stress function resulting in eqn. 10.

$$\frac{\partial^4 \phi}{\partial x_1^4} + 2\frac{\partial^4 \phi}{\partial x_1^2 \partial x_2^2} + \frac{\partial^4 \phi}{\partial x_2^4} = 0$$

where

$$\sigma_{11} = \frac{\partial^2 \phi}{\partial x_2^2}, \quad \sigma_{22} = \frac{\partial^2 \phi}{\partial x_1^2}, \quad \sigma_{12} = -\frac{\partial^2 \phi}{\partial x_1 \partial x_2}$$

eqn. 10

Requiring the selection of a function, ϕ, satisfying known boundary and loading conditions greatly restricts the complexity of component geometry, which must then

be overlaid by simplistic forms. Though such functions can be relatively straightforward their application to a strength of materials theory is extremely arduous and normally outside the scope of the typical engineer and in the domain of the mathematician. However, up until the last few decades analytical solutions were the only real method by which to study the mechanics of a system. In recent years numerical approaches have been developed in an attempt to bypass this problem, resulting in the methods of finite difference (FD) and finite element (FE).

Both methods are long accepted laborious approximations to the differential and integral equations rising from the theory of elasticity. It took the advent of the computer in the 1960's to start further interest in these methods. Probably the best illustrative example of the difficulties inherent in the FD method is that of Benham and Hoyle [78]. An autoclave blocking ring is modelled through FD approximations of the elastic equilibrium equation and the mathematics presented is just as arduous as that of an analytical approach. Once equations have been determined a very specific iterative routine needs to be programmed and correctly maintained during running by an operator who must adjust values in accordance with a hand drawn graph of the programmes results.

An earlier example of applying FD to thermomechanical problems is that of Boley [79], a thorough analysis is given on a thick bar with a thermal distribution. Though a two dimensional case is proposed only a one-dimensional stress system is given. It is clear from both these examples that the FD approach to mechanical solutions is not far removed from the specific complexity of analytical solutions.

Although complex for approximations to high order mechanical theory the FD method is still a straight forward method for approximating the low order heat transfer solutions and can give equally good results as those of an expensive FE package. This will be seen in later sections on the work conducted.

FE was not a consequence of attempts to work around the FD complexities, both methods developed together, though significantly more evolution took place in FE. In 1976 Brown [80] conducted work on expanding FD methods into elastic-plastic studies still showing itself to be highly mathematically intensive. Whilst much earlier in 1960 Clough [81] presented a paper describing the manner of elastic plane stress analysis as it exists today in commercial FE codes. However one may find the beginnings of FE in 1941 by Hrennikoff [82] where a component was replaced by a wire mesh framework of struts organised to a specific pattern for optimum

performance. This is reminiscent of the geometry requirements posed in finite element for an appropriate solution to be gained.

Much work was conducted in the 1970's to expand the applicability of FE to fracture mechanics and life prediction whilst also using it as a tool to test the then standard recommendations of fracture toughness testing. Efforts were also made to further understanding of various crack tip parameters such as crack opening displacements (COD) and Rice's path independent integral, J. To this effect the Department of Industry issued a report [83] on the determination of stress intensity factors in single edge-cracked plates using FE. Various methods are presented and compared to a collocation solution of which J was the recommended, though not the easiest or quickest. Once determined, J was readily converted to K via G equivalence.

In 1977, Atluri *et al* [84] used a special embedded singular element type to model the presence of a blunt crack tip in a ductile material. Using FE the effects of incremental plasticity, kinematic hardening and non-linear geometry were all included into the model, excellent correlation to experimental results was found. It was also determined that the path independence of J demonstrated good independence under linear-elastic conditions and though slightly reduced in the presence of plasticity still maintained good independence. This slight reduction in path independence under elastic-plastic conditions is still a consideration in contemporary codes such as ABAQUS.

In the same year using FE, Roche [85] suggested that the calculation of J for elastic-plastic materials be best determined by the incremental plasticity method as opposed to the deformation plasticity method. This however, is likely due to the behaviour of the code used, and probably leant more towards solving for linear response models. Parks [86] also demonstrated that a method for calculating J based on energy comparison of two slightly different crack lengths can be replaced by a single elastic-plastic FE analysis where the crack tip position is altered slightly. This method was successfully applied to 3D analysis. In 1998 Yuan *et al* [87] showed that J provided a good estimate of stress intensity factors under thermal transient loading. However, the method used reduced in accuracy with increasing crack lengths.

The nodal displacement nature of FE immediately lends itself to the concepts of COD. During this period Kuo *et al* [88] , Nakagaki *et al* [89], Dawes [90] and Berger *et al* [91] studied crack tip behaviour in relation to COD. Whilst not using FE, Berger used various methods to determine the fracture toughness of steels with both linear-

elastic and elastic-plastic methods. Demonstrating that established methods of experimentally determined stress intensity factors by CTOD measurement compared very well with J-Integral methods shows the importance of the COD in research of this time. Kuo calculates COD by FE and determines by experiment under various thickness of specimen. Kuo suggests that the standard Dugdale model is inappropriate for thick specimens at high loads under which condition elastic-plastic FE is more preferable. Bilby [92] conducted similar work in the late 1980's where it is suggested size limitations on fracture toughness parameters may be inadequate for lightly hardening materials.

Studying the effects of size on fracture toughness, by or supported by, the FE method was taken a step further by attempts to model crack growth. Nakagaki *et al* in 1979 constructed a model of the stable crack growth of a centre cracked specimen under uniaxial loading. The results were matched admirably to those of experimentally determined data. However, in order to advance the crack length incrementally, constant increases were used resulting in a difficulty to determine the onset of unstable crack growth. Similar to this was work on solder joints by Ju *et al* [93] in 1993, mixed mode crack growth was addressed with similar methods by Reimers [94] and complex load histories were introduced by Jeng *et al* [95]. Here the crack path was predetermined and ΔJ-Paris law correlation was used again with constant increases in crack length. A good comparison to experimental results was found, however the model normally requires excessive input from these experimental results, reducing the independence of the model. This form of analysis generally marks the start of continuum damage mechanics (CDM), whereby the introduction of experimentally determined material properties such as softening ahead of the crack tip to generate void growth are introduced to model the actual damage of the material on a macroscopic scale.

Some examples of CDM are given by Ayari *et al* [96], Hayhurst [97] and Chow *et al* [98]. Ayari in 1994 and in Hayhurst 1996 introduce creep fatigue interactions with Ayari demonstrating the parameter intensive nature of CDM. In this case a methodology is set-up rather than comparing model performance to experimental results and uses 18 different parameters. Chow introduced a unified method in 1997 where the elastic energy release rate was combined with damage mechanics theory allowing crack propagation to be modelled as a function of microcrack nucleation. The

results however, were not promising, producing similar form results, with errors from 3% to 86%. Damage mechanics had already been applied by Bilby *et al* [99] in 1995, where the method of cell softening to modelling void nucleation ahead of a crack tip is offered. Results presented are based on cylindrical specimens and show compelling performance of these models. In 1997 the model was further refined [100]. However, conclusions were drawn that further geometries needed to be tested in order to fully validate the model.

Today FE theory is widely accepted and used in industry for a huge array of problems in areas of automobile suspension, chassis design and static, thermal, elastic and elastic-plastic stress analysis. Though the theory of FE is complex it possesses a foundation that is consistent across all engineering problems and it is this which makes it capable of programming into a versatile tool for modern engineers to advance their capabilities.

As mentioned previously, Hellen and Chen showed determination of K values from J-Integrals to be very thorough, however, further methods are available under LE conditions. Shown also by Chen, a similar method is shown by Chu [101] where displacements near a crack tip can be used to determine stress intensity factors. This requires the use of an elegant manipulation of the FE mesh to incorporate the singular nature of the crack tip.

2.2.2 – Crack Tip Finite Elements

The versatility of FE inevitably grew into the application to fracture mechanics. In 1971 Tracey [102] described the use of a new finite element called a *triangular singularity element*. In truth this was a four noded quadrilateral element with two adjacent points collapsed together as in Figure 11a. A specific shape function was then applied to match the behaviour of standard stress intensity functions and the element surrounded by standard quadrilateral elements. A 5% error was determined from models when compared to theoretical values. In 1975 and 1976, however, Henshell & Shaw [103] and Barsoum [104] respectively removed the need for such separate calculations for modelling a crack tip singularity. Using standard eight noded quadrilaterals, as in Figure 11b, midside nodes adjacent to a crack tip were displaced to a quarter distance of the element side from the crack tip instead of half way along.

From these *quarter-point* elements it was found Westegaard's inverse square root of distance from the crack tip resulted in the element formulation. Barsoum also touches on the possibility of collapsing two adjacent nodes to the crack tip to create a degenerated triangular element as Tracey suggested. This method is present today in the commercial FE software ABAQUS and incorporated into most custom written fracture mechanics FE codes.

The two or even three-dimensional nature of the quarter-point element provides the ability to model full mixed mode loading and is presented in both the Henshell and Barsoum publications. Double edge notch, three-point bending, and centre cracked specimens all result in very good performances when compared to the well established classical solutions. Further to this Saouma [105] conducted extensive numerical studies on these elements to present the following recommendations for their use:

- Use at least 4 quarter point elements around the crack tip for mode I loading and 8 for mode II loading.
- Use a reduced 2x2 integration scheme on quarter point elements.
- Keep internal angles of triangular quarter point elements approximately equal to 45°.

2.2.3 – Numerical Thermal Shock Modelling

Probably the most recent and best analytical solution to a thermally shocked edge cracked plate is that of Nied in 1987 [106, 107]. Though analytically determined the final equations still had to be numerically solved for a solution. Conducting analysis on both downshock and upshock Nied determined the relationship between Fourier number and maximum stress intensity through the cycle. Interestingly he also determined the tendency of crack closure at the surface during an upshock cycle generating a form of internal cusp shaped crack. He concludes that the normally less severe upshock stress intensities can become larger than the downshock stress intensities at approximately one third through the width of the plate. Therefore in combined upshock and downshock cycles crack retardation in downshock compressive regions could be over ridden by upshock stresses.

A similar analytical study leading to a numerical solution was conducted some time later in 1993 by El-Fattah *et al* [108, 109]. Still relating all shocks to the Fourier number El-Fattah addressed the non-linear rapid cooling and heating of thermal

shocks with both a step and ramp function, concluding the instantaneous change in temperature caused by a step function resulted in the severest of stresses.

Other numerical analyses of note for the thermal shock problem are those by Xuejun *et al* [110], Tsai *et al* [111] and Lee [112]. Xuejun studied a surface crack subject to thermal loading whilst Tsai and Lee studied the concepts of weighting function methods for edge crack problems. Shimamura [113] also attempted to model the effects of craze-cracking using a 3D lattice propagating cracks based on a strain energy basis. Models showed near independence on the lattice shape though it was only applied to LiF crystals. Numerical simulation of thermal shock alumina disks was also conducted by Tomba *et al* [114,115], where thermal shock was achieved by the use a high velocity air jet, as used previously by Hasan. The work was centred on determining the thermal shock resistance of various surface finishes to crack initiation with experimental results explained on the basis of FE stresses.

Kokini [116] presented FE thermal shock models for edge-cracked plates and internally cracked plates in plane strain. Using a numerical equation to determine K_I values and comparing to the analytically determined K values by Nied, Kokini found a very good correlation.

Work into the mixed mode nature of thermal shock has been conducted by Chen *et al* [117] and Katsareas *et al* [118]. A consideration of K_{II} is unusual, as models were symmetrical about the crack plane as was loading, except Chen who conducted inclined internal crack models, an issue not addressed by Nied, Rizk or Lee. Though Katsareas' work was based on a boundary-only element method, both works determined K_{II} values approximately an order less than K_I values. This is supported by the work of Emery in 1977 [119] who used the quarter point finite elements to determine a solution to edge cracked thermal downshock.

2.3 – Mixed Mode Loading

A short section on mixed mode loading is presented here as it can have some bearing on thermal shock. Mixed mode loading is a relatively new area and little understood to the extent of standard applications. Its importance comes from the ability of mixed mode scenarios to increase crack growth rates significantly enough to render single mode tests inappropriate. The main difficulty comes from the inability of

different mode stress intensities to be summed to an overall effective K value and the lack of an otherwise sufficiently directional crack tip parameter. The separate mechanical modes are shown in Figure 12.

Mixed mode scenarios by thermal loading are also possible. Emery [120] determined a K_I and K_{II} analysis of thermal shock as caused by severe thermal gradients existent at the crack tip. These gradients can be considered as a direction of heat flow not parallel to the cracked surfaces. Under such conditions heat flow would be obstructed by a break in material behind the crack tip and continuous ahead of the crack tip as illustrated by Figure 13 resulting in a high thermal gradient across the crack tip, i.e. a thermal singularity. Just how such a scenario is quantified is never made clear by Emery, instead a thermal singularity value is determined and used as an additional term to K_I and K_{II} equations. Since Sih, section 2.1.2, showed stress intensities could be determined with thermoelastic stresses the classical descriptions of stress at a crack tip can still be applied. The descriptions of stress components at a mixed mode loaded crack are given by LEFM [4] as eqn. 11

$$\sigma_{xx} = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right] - \frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left[2+\cos\frac{\theta}{2}\cos\frac{3\theta}{2}\right]$$

$$\sigma_{yy} = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1+\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right] + \frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\cos\frac{\theta}{2}\cos\frac{3\theta}{2}$$

$$\sigma_{xy} = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\sin\frac{\theta}{2}\cos\frac{3\theta}{2} + \frac{K_{II}}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]$$

eqn. 11

For mechanical loading isothermal biaxiality and mixed mode loading are not necessarily mutually combined. Only when a crack rotates or branches to a non-orthogonal angle to applied loads, can it be certain that mixed mode loading is taking place. In thermal loading mixed mode conditions will arise if a non-zero angle is present between the crack and direction of heat flow as described above and by Figure 13.

Quantification of mixed mode loading has been attempted with some success by Hua *et al* [121] using the plastic zone size in the Paris law.

$$\frac{da}{dN} = \frac{14}{0.8066}\left[\frac{\sigma_{yield}}{E}\right]^2 r_P'$$

eqn. 12

Where r_P' is the maximum radius of the plastic zone and given as a function of both K_I and K_{II}. The correlation at larger plastic zones, around 300μm, showed a good fit, though for smaller plastic zones the correlation could over estimate by some 50%.

Pook [122] attempted to characterise combined mode I and III loading with little success regarding thin plates. Hua and Fernando [123], however, continued to address the plastic zone size in non-proportional overloading finding again the relevance of plastic zone size and shape with regard to its component K_I and K_{II} forms. The plastic zone therefore seems to be an obvious candidate for quantifying the effects of mixed mode loading. Using a von Mises yield criterion, Mohr's stress circle and eqn. 11 the plastic zone size can be determined for both plane stress and plane strain as eqn. 13 and eqn. 14 respectively.

Plane Stress

$$r_P = \frac{1}{2\pi\sigma_{YS}^2}\left\{\begin{array}{l} K_I^2\cos^2\frac{\theta}{2}\left[3\sin^2\frac{\theta}{2}+1\right] \\ +K_I K_{II}\sin\theta[3\cos\theta-1] \\ +K_{II}^2\left[3+\sin^2\frac{\theta}{2}\left(1-9\cos^2\frac{\theta}{2}\right)\right] \end{array}\right\}$$

eqn. 13

Plane Strain

$$r_P = \frac{1}{2\pi\sigma_{YS}^2}\left\{\begin{array}{l} K_I^2\cos^2\frac{\theta}{2}\left[3\sin^2\frac{\theta}{2}+(1-2\nu)^2\right] \\ +K_I K_{II}\sin\theta\left[3\cos\theta-(1-2\nu)^2\right] \\ +K_{II}^2\left[3+\sin^2\frac{\theta}{2}\left((1-2\nu)^2-9\cos^2\frac{\theta}{2}\right)\right] \end{array}\right\}$$

eqn. 14

Resulting plastic zones are displayed in Figure 14. The consequence of mixed mode can then be seen to rotate the plastic zone about the crack tip, effectively altering the angle of shear planes. The result is a rotating crack.

2.4 – Thermal Shock Biaxiality

Thermal shock is a severe form of thermal loading, generating low cycle fatigue lives. Biaxiality is an additional component to uniaxial loading which is known to reduce fatigue lives under isothermal conditions, but can also be generated by multi-dimensional thermal loading. It is important therefore that the presence of biaxiality in thermal shock be quantified. To this date very little work has been done on the subject, specifically the work by Bass *et al* [77] previously mentioned in section 2.1.6 who found a large reduction in fracture toughness is a result of mechanical biaxial loading on surface flaws.

Since mechanical biaxiality in specimens is a result of applied far-field loads, the biaxial stress field present is consistent over a broad region. Biaxiality generated under thermal loading will not reflect this uniform distribution since it is thermal gradients that create stresses. Therefore it is surmised that bounding limits of biaxiality will be present under thermal loading conditions and these limits could be reproduced via uniform mechanical stress fields. Should these fields be generated under isothermal conditions at the maximum temperature of the thermal shock cycle crack growth rates could be related to those of thermal shock.

It is this problem that the following work is concerned with. By using isothermal crack growth rates at limiting cases of biaxiality and thermal shock crack growth rates, a correlating analysis of the two is presented. From the results, it will be shown whether the effects of isothermal biaxiality can be compared with the effects of biaxiality from thermal loading.

3 – Methodology

3.1 – General Information

Two types of numerical analyses were conducted; finite difference models to determine applied forced convection loads and finite element models to determine two dimensional thermal and stress distributions.

Finite difference models were written using a Borland C compiler version 4.52 and run on a standard PIII 450MHz PC. Finite element models were constructed through ABAQUS Standard version 5.7 using ABAQUS Post version 5.7-7 for visualisation of results. A Sun Sparc 2 workstation was used with a remote connection to a Sun Microsystems Enterprise E450 server with dual 248MHz processors and a maximum available memory of 1Gb and user quota of 5.5Gb.

Pre-processor user programmes, data extraction programmes and ABAQUS user routines were written with a Sun Microsystems Workshop containing C compilers and Fortran 77 version 5.

A general methodology for numerical analysis can be performed as follows:

1: **Conceptual modelling** - determine environmental constraints: thermal, fluidic, mechanical and/or dynamic loads for theoretical model. Locate any planes of symmetry and evaluate material properties for specific model type, thermal, fluidic, mechanical or dynamic.

2: **Pre-Processing** - generate geometry and refine model to appropriate accuracy, observing characteristics such as element density for use in further analyses of a similar kind.

3: **Processing** - run analysis using appropriate platform; commercially available software or user code.

4: **Post-Processing** - extract required information.

For the thermal shock models conducted in this work a more detailed methodology is presented, isothermal biaxial models are conducted in much the same manner with the obvious exception of thermal considerations.

3.2 – Modelling Methodology

3.2.1 – Thermal Shock Model

A thermal stress analysis was constructed through a two stage uncoupled superposition of thermal distribution to a mechanically constrained model. The thermal analysis consisting of a two-stage cycle was first edge cooled from a maximum initial temperature to a final end temperature using a film convection loading routine allowing specification of ambient conditions and forced convection coefficient (*surface heat transfer coefficient, h, $W/m^2.K$*). A value of *h* was determined for the required thermal gradient and thermal shock cycle via a finite difference programme written in C (appendix B.1.). This programme was developed for both non-temperature dependant and temperature dependant thermal properties using modified Fourier numbers to determine the required *h* value through a linear iteration routine. The thermal distribution was then passed into its second stage reheat cycle to the initial maximum temperature via a user routine (appendix B.2) reducing the current thermal gradient at a node to zero at a decaying exponential rate.

This thermal distribution was then imported to a static stress model of the same design with mechanical constraints applied as shown by Figure 15. A symmetry condition was applied at *AB* and a single node zero x-displacement condition at *C* situated centrally through the width. This provides a full two dimensional description of restraint whilst still allowing free bending to take place under non-symmetrical thermal loading.

Model geometry was kept as simple as possible whilst maintaining the same fixed points as specimens tested in the previous study [124] for use of crack growth data. This required a minimum height of 106mm from the symmetry plane *AB* to the central pin jointed mechanical loads situated at point *C* in Figure 15. A plot of the finite element mesh is shown in Figure 16a and b and the forms of loading in the models conducted are shown in Table 1.

3.2.2 – Cruciform Model

Isothermal-biaxial stress analyses were conducted under limiting cases of biaxiality. Sinusoidal stress ranges of 118MPa and 200MPa with an R-ratio of 0.1

were applied perpendicular to the crack, σ_N. Applied parallel to the crack both dead loads and sinusoidal loads, σ_L, were applied at the mean level of σ_N for unibiaxial loading and equal to σ_N for equibiaxial loading. A plot of the finite element mesh used is shown in Figure 17a and b and loads conducted tabulated in Table 2.

3.3 – Finite Element Pre-Processing

Standard two dimensional eight noded continuum elements for plane stress were used in the analyses. This allowed the inclusion of a singularity effect into the model at the symmetry plane through quarter point element manipulation, with degeneration of quadrilateral elements into triangular elements surrounding the crack tip position. In stress models this resulted in many free nodes occupying the same point at the crack tip which could not be relied upon to accurately open the crack under application of thermal loading, therefore further constraints were required. The y-displacements of these multiple nodes at the symmetry plane was fixed in the same way as other boundary conditions. For the remaining active degree of freedom, x, one of the multiple nodes was allowed to displace freely along the symmetry plane with displacements of the remaining multiple nodes equated to any displacements at this free node.

It was found for a model width of 40mm a crack tip singularity mesh of 1mm square containing over 50 elements was sufficient. This small size allowed relatively small edge cracks to be modelled whilst maintaining 6 or more element paths around the crack tip for a suitable averaging of the J-Integral crack tip parameter output by the model. Crack tip stress refinement was difficult to obtain, however the above allowed stresses one or two nodes ahead of the crack tip to refine well.

These refinement characteristics were written into a C programme to calculate mesh-numbering specifications allowing changes in crack length, element density and biasing within an element aspect ratio of 12 to 1 (appendix B.3, appendix B.4). A second programme then used these specifications to generate the required number of ABAQUS input files for processing (appendix B.5, appendix B.6). Sample input files are listed in appendix B.7 and B.8 for thermal shock and cruciform models respectively.

3.4 – Finite Difference Processing

The finite difference model to determine heat transfer coefficients is a linear estimation to the non-linear thermal distribution. As is shown in later chapters the solution is very close to theoretical and finite element results. The programme runs in minutes and executes in an MS-DOS prompt window.

A programme layout is as follows,

- *User Input*
  - Uniform start temperature
  - Final surface temperature
  - Ambient temperature
  - Model length & Element quantity
  - Material properties (bulk)
    - Thermal Conductivity
    - Density
    - Specific Heat
  - Initial surface heat transfer coefficient
  - Time Period
  - Output to Summary File Setting
- *Output File Setup*
  - Setup summary file layout
- *Heat transfer iteration loop*
  - Reset/write initial temperatures to array 1
  - *Time loop*
    - *Node loop*
      - Scan relevant nodal temperatures from array 1 for calculating new temperature
      - Calculate new nodal temperature, Select appropriate model equation
      - Output new temperature to array 2
      - Output new temperature to summary file if required
      - Increment Node number
    - *End Node loop on last node condition*
    - Copy all written temperatures from array 2 to array 1
    - Increment Time by one time increment
  - *End Time loop on end of time condition*
  - Output heat transfer iteration summary to summary file
  - Test final surface temperature
  - Apply error to heat transfer coefficient
  - Re-calculate time increment and set to a suitable value
- *End heat transfer iteration loop on final temperature condition*
- *Close down files*

A complete solution is required for every time increment conducted. Therefore, to save memory on the many time increments required over a transient solution, two solution arrays are constructed. The programme then alternates between these two arrays writing any complete solutions through the time period required to a text file.

Once all information is collected from the user, the programme runs through time incrementally, according to a minimum time increment. The temperature at each node is calculated based on the previous time increment and the appropriate equation for that location in the model. When all nodal temperatures have been calculated, the time incrementation proceeds by one increment and the process begins again until the end of the time period specified by the user is reached. The end result is checked against the required temperature, specified by the user, repeating if not within tolerance or ending if within tolerance and writing all information to a summary file for post-processing.

In order to find the required heat transfer coefficient a simple linear iteration scheme is used where an error from the required result is fed back into the previous heat transfer coefficient until that error drops below 0.1%.

3.5 – Finite Element Processing

Analyses were run with a remote server connection taking approximately 3 hours for the thermal-stress uncoupled analyses to complete. Isothermal biaxial models would take significantly longer, around 4 to 5 hours due to increased mesh size.

A reheat cycle was obtained through a simple inverse exponential equation given as eqn. 15 (appendix B.2),

$$T = T_{MIN} + \Delta T\left[1 - e^{-kt}\right]$$

eqn. 15

where ΔT, k and t are the thermal gradient, a constant of decay and reheat time period respectively. A single constant of decay was determined by comparison to a previous study on thermal distribution tests, see Table 3 [124], and used to describe reheat magnitudes. This is required since the rate of cooling and reheat is believed to affect the magnitude of end stresses, and so any estimation of ΔK_I or other crack tip descriptor. Consequently a linear or step reheat would be insufficient.

3.6 – Finite Element Post-Processing

Post-processing was conducted using ABAQUS' own post-processor ABAQUS Post. This requires the generation of a coded results file in excess of 100Mb. Thermal and stress distributions, crack opening displacements and J-Integral values are easily output to file, however recording of plastic zone size and shape is less convenient and inaccurate. By plotting the stress distribution to a limiting value, i.e. the yield stress, the plastic zones are easily observed. However, recording the contours of the zone perimeter required estimating distances between nodes all the way around the perimeter. This leads to cumulative errors and consumes much post-processing time for every analysis. To ease this, a Fortran user routine (appendix B.9) and data extraction programme (appendix B.10) was written in C. The user routine writes yield zone perimeter stresses with element numbers and integration points to the ABAQUS data file. The data extraction programme then searches the data file converting the user routine information in to x-y co-ordinates by correlating element numbers to node numbers and their corresponding x-y co-ordinates. This then defines the plastic zone perimeter and is then tabulated to a text file.

Since these perimeter stresses are only available at integration points within elements, the programme must search for co-ordinates of nodes defining these elements, determining integration point locations to interpolate for the two yield points as shown by points A and B in Figure 18.

3.7 – Evaluation

This method is repeated for different crack lengths to obtain a crack tip descriptor for any severity of thermal downshock loading or isothermal biaxial scenario. Beyond a crack length of 5mm gross increases in plastic zone are observed, however stress intensities determined by eqn. 16 are found to remain fairly consistent with theory.

$$K_I = \sqrt{JE}$$

eqn. 16

When coupled with crack growth data this method can be used to determine different crack tip descriptors for comparison to isothermal biaxial crack growth rates.

4 – Materials Analysis

4.1 – Property Evaluation

4.1.1 – A Ramberg-Osgood Analysis

Cyclic material properties have been obtained for solution annealed AISI 316 stainless steel [6] for room and elevated temperatures and presented in Table 4. For elastic analyses a cyclic modulus and 0.2% proof stress are plotted in Figure 19. It is immediately clear that little changes in cyclic modulus are encountered at elevated temperatures. However, a slight discontinuity exists at around 500°C as a result of material hardening from precipitation of carbides. Rios *et al* [38] described this as typically coarse carbides at the grain boundaries of 0.05 to 0.1μm and fine transgranular carbides of 0.01-0.02μm. This is believed to induce the large changes in 0.2% proof stress as a consequence of relatively large changes in strain hardening exponent.

Using a Ramberg-Osgood model in the form of eqn. 17,

$$\varepsilon = \frac{\sigma}{E} + \left(\frac{\sigma}{K}\right)^n$$

eqn. 17

for a full elastic-plastic (EP) consideration these large changes in strain hardening caused by carbide precipitation are illustrated by Figure 20. Failure stress is not consistent with temperature, mirroring changes at 500°C and 700°C as with cyclic modulus, although over all ductility does increase with temperature.

For finite element models considering only the downshock cycle unloading is not a consideration. Therefore the non-linear elastic constitutive law *deformation plasticity* can be used and written as eqn. 18.

$$\varepsilon E = \sigma + y_o \left(\left|\frac{\sigma}{\sigma_{yield}}\right|\right)^{n-1} \sigma$$

eqn. 18

Material constants for a stress-strain response equivalent to eqn. 17 are given in Table 5.

4.2 – Bulk Thermal Stress

4.2.1 – Incremental Plasticity

A bulk thermal stress represents a general stress from a uniform thermal distribution. For an elastic analysis this is obtained simply through an interpretation of Hookes' law for thermal strain by eqn. 19.

$$\sigma_{TH} = E\alpha T$$

eqn. 19

This is used by Timshenko's theoretical elastic thermal stress equation. Since thermal stresses by this method will be excessively over yield it is appropriate to model plasticity into the calculation. Using the material properties given in Table 4 and eqn. 17 an equivalent EP bulk stress can be determined with a Newton-Raphson iteration procedure. Rewriting eqn. 17 as eqn. 20 for known thermal strains

$$\frac{\sigma}{E}+\left(\frac{\sigma}{K}\right)^{n}-\alpha\Delta T=0$$

eqn. 20

the Newton-Raphson iteration can be performed using eqn. 21.

$$\sigma_{i+1}=\sigma_i-\frac{\frac{\sigma_i}{E}+\left(\frac{\sigma_i}{K}\right)^{n}-\alpha\Delta T}{\frac{1}{E}+\left(\frac{\sigma_i^{n-1}}{K^{n}}\right)(n-1)}$$

eqn. 21

Introduction of $\alpha\Delta T$ as opposed to αT provides a full elastic-plastic consideration. In elastic analyses Timoshenko produces mathematically equal results for both αT and $\alpha\Delta T$, however when plasticity is introduced this equality ceases to exist. This problem is demonstrated in a later section.

Four material models are conducted and illustrated in Figure 21 and Figure 22.

1: Cyclic linear elasticity
2: Elastic-Plastic full Ramberg-Osgood
3: Pseudo mechanical elasticity
4: Linear-Elastic (mechanical) Elastic-Plastic (Ramberg-Osgood)

Figure 21 shows models 1 and 2 where a full Ramberg-Osgood relationship is calculated through eqn. 21 and compared to the cyclic modulus response. Figure 22 shows models 3 and 4 where a mechanical modulus is determined from a 0.2% proof

stress and equivalent strain. The constitutive law used to obtain a thermal bulk stress is determined by the following simple conditions,

$$\text{if} \quad \alpha\Delta T \leq \varepsilon_{\text{yield}}$$

then use eqn. 19 with mechanical modulus

$$\text{if} \quad \alpha\Delta T > \varepsilon_{\text{yield}}$$

then use eqn. 21 with Newton-Raphson method

where $\varepsilon_{\text{yield}}$ is determined by

$$\varepsilon_{yield} = \frac{\sigma_{PS}}{E} + \left(\frac{\sigma_{PS}}{K}\right)^{n}$$

$$\text{where} \qquad \sigma_{PS} = K(0.002)^{1/n}$$

$$\text{and mechanical modulus} = \frac{\sigma_{PS}}{\varepsilon_{yield}}$$

The two pairs of models illustrate large differences as a consequence of the high ductility of AISI 316. The difference in strain energy of both LE and EP models for both pairs is significant. Figure 22 shows the least change in strain energy as a consequence of using plasticity and therefore little difference in the final solution should be determined. However, such a material model would grossly underestimate the non-linear stresses occurring in the mechanically linear region as shown by the Ramberg-Osgood line. The consequence of larger differences in strain energy should therefore show larger deviations of the final non-linear solution to the linear solution. This will be shown in a later section.

5 – Theory

5.1 – Analytical Solution

5.1.1 – Crack Growth Data

5.1.1.1 – SECANT METHOD

Crack growth data for single and double edge thermal downshock on a thin stainless steel plate and isothermal-biaxial cruciform specimens from previous work [124] are presented in Table 6 through Table 13. It is required that an account is made of how such data is evaluated. The ASTM standard E647 [125] provides two methods of evaluation, the secant method and the incremental polynomial method.

The secant method is a simple forward difference calculation of the slope between two points of the *a-N* curve and is given in eqn. 22

$$\left(\frac{da}{dN}\right)_{\bar{a}} \approx \frac{a_{i+1} - a_i}{N_{i+1} - N_i}$$

eqn. 22

where i is a sequential data point number. Such a method works well with relatively simple data distributions.

5.1.1.2 – INCREMENTAL POLYNOMIAL METHOD

The incremental polynomial method allows a short sequential series of points to be fitted with a second order polynomial shown by eqn. 23.

$$a_i = b_0 + b_1\left(\frac{N_i + C_1}{C_2}\right) + b_2\left(\frac{N_i + C_1}{C_2}\right)^2$$

$$-1 \le \left(\frac{N_i + C_1}{C_2}\right) \le +1$$

eqn. 23

This allows some elimination of the inevitable scatter found in the data. Unlike the secant method laborious calculations are required possibly using a least squares iterative solution. However, the standard provides a Fortran code listing to evaluate the

data. An adaptation of this routine for coding into a Matlab solver is listed in appendix B.11.

5.1.2 – Isothermal Biaxiality

5.1.2.1 – T-STRESS

Isothermal biaxiality has been discussed previously in section 2.1.2. The importance of T-stress is emphasised by Brown and Miller [31] and Kfouri and Miller [26,30] using the cruciform specimen. Defined in full the non-singular term for biaxiality around a crack tip is written as eqn. 24 and should not be confused with σ_T of eqn. 5 and 6.

$$TStress = \sigma_N (1 - \Lambda)\cos 2\alpha$$

eqn. 24

This is illustrated by Figure 23, where α is the angle spanned from the normal axis to the crack plane. Under mode I conditions $cos2\alpha = -1$, therefore the description simplifies to

$$TStress = \sigma_L - \sigma_N = \sigma_T$$

eqn. 25

Kitagama [32] presented similar work on slant cracks using a biaxial specimen differing by the presence of large in plane radii. The work determined biaxiality has little or no effect at low stresses and long cracks whilst influencing crack growth at short cracks and high stresses. This was explained by the theoretical work suggesting that fully tensile biaxial loading creates smaller plastic zones than tension/compression biaxial loading. Consequently at lower stress levels tension/compression loading can generate higher crack growth rates than fully tensile loading. Crack orientation was also found to occur such that K_I reached a maximum and K_{II} a minimum.

Rhodes and Radon [34,126] presented a simple conclusion on biaxiality as the tendency for centre cracked specimens to have higher crack growth rates than those of standard CT specimens.

5.1.2.2 – STRESS INTENSITY FACTOR

To determine the behaviour of a crack tip under biaxial loading the Brown and Miller cruciform specimen design was used. Using a standard centre cracked specimen geometry, fins are added to all four sides such that a load can be applied distant from the plate edges allowing a form of free deformation. Applying an orthogonal dead load can counteract any distortion caused by Poisson's ratio. The specimen geometry is sufficiently complex however, to require a more in depth study. Brown and Miller [31] present a correlation of fatigue crack growth rate with ΔK as defined by eqn. 26.

$$\Delta K = \Delta\sigma \sqrt{\frac{\pi a}{\cos\left(\frac{\pi a}{W}\right)}}$$

eqn. 26

Finite element models of the cruciform specimen conducted in this work have shown that variable thickness between the plate and loading fins is a governing factor. Consequently the strength of in and out of plane stress raisers at the junction of plate to fins is likely to cause problems. Plane stress variable thickness models are problematic whereas uniform thickness equated to that of the plate are more in line with eqn. 26.

5.1.3 – Biaxial Plastic Zone

5.1.3.1 – COMPONENT STRESSES

Plastic zones for stresses normal and lateral to the crack plane can be determined simply from eqn. 11. Both normal, σ_{yy}, and lateral, σ_{xx}, stresses can be plotted to determine plastic zone sizes and shapes and are presented in Figure 24. However, only at low stresses are both lateral and normal plastic zones of similar radius at $\theta = 0$. In order to allow the lateral plastic zone size to differ from the normal zone size the T-stress term must be accounted for in the lateral stress radii, σ_{xx}. Singular stresses at a crack tip are now written as eqn. 27 and illustrated by the small lateral plastic zone of Figure 24. Appendix A shows how a von Mises plastic zone accounting for T-Stress is determined.

$$\sigma_{xx} = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right] - \frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left[2+\cos\frac{\theta}{2}\cos\frac{3\theta}{2}\right] + \sigma(1-\Lambda)\cos 2\alpha$$

$$\sigma_{yy} = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1+\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right] + \frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\cos\frac{\theta}{2}\cos\frac{3\theta}{2}$$

$$\sigma_{yy} = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\sin\frac{\theta}{2}\cos\frac{3\theta}{2} + \frac{K_{II}}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]$$

eqn. 27

5.1.4 – Thermodynamics

5.1.4.1 – THEORETICAL ONE DIMENSIONAL DESCRIPTION

Thermodynamics can be separated into the transient and steady state forms of heat flow. All dynamic systems begin with some transient behaviour where transient heat flow is defined by the rate of change of the thermal gradient between two points through time. A more practical description would be that the shape and magnitude of a transient thermal distribution curve changes through time. When the thermal gradient ceases to change appreciably through time the transient state is said to have settled into steady-state conditions. The distribution curve will then only change in magnitude. Both transient and steady-state forms for a one dimensional consideration are represented by eqn. 28 and eqn. 29 respectively, where T, x and t are the temperature, distance and time respectively.

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{D}\frac{\partial T}{\partial t}$$

eqn. 28

$$\frac{\partial^2 T}{\partial x^2} = 0$$

eqn. 29

D is the *thermal diffusivity* and is given by eqn. 30, where k is the thermal conductivity, ρ is the density and C_P is the specific heat capacity.

$$D = \frac{k}{\rho C_P}$$

eqn. 30

Thermal shock is a transient heat transfer process and so steady-state will play no part in its analysis. Figure 25 shows the straight forward scenario of eqn. 28, a single edge cooled plate, one edge is considered perfectly insulated and the opposite edge cooled by free or forced convection. A solution to eqn. 28 can be found in Nied [106, 107] and Eastop and McConkey [127] as derived by the boundary conditions,

$$\text{at } t = 0 \Rightarrow \Delta T = T - T_A$$

for initial uniform temperature conditions, T_A = ambient temperature

$$\text{at } x = 0 \Rightarrow \frac{\partial T}{\partial x} = 0$$

at the insulated surface, $x = 0$, no heat flow occurs

$$\text{at } x = L \Rightarrow k\frac{\partial T}{\partial x} = h[T_A - T]$$

for heat flow at the convecting surface

For this simple one dimensional problem the analytical solution is given as eqn. 31 where, L is the length of the model, T_I is the initial condition temperature and Fo is the *Fourier number.*

$$\frac{T - T_A}{T_I - T_A} = 2\sum_{n=1}^{\infty}\left[\frac{\sin(p_n L)\cos(p_n x)}{p_n L + 0.5\sin(2p_n L)}\right]e^{-(p_n L)^2 Fo}$$

$$Fo = \frac{kt}{\rho C_P L^2}$$

eqn. 31

The values p_n are eigenvalues determined from eqn. 32 where Bi is the *Biot number* and h is the surface heat transfer coefficient.

$$(p_n L)\tan(p_n L) = Bi = \frac{hL}{k}$$

eqn. 32

5.1.4.2 – EMPIRICAL SOLUTION

Further to the analytical solution, Hasan presented an empirical solution based on thermal downshock experiments [64]. The theory correlated well to experimental results and is used as a function of distance and time to determine the thermal distribution.

$$Z<0:\ T = T_{\max}(1-At) - T'\left(\frac{0.3+1.70t}{1+1.70t}\right)Z^4$$

$$Z\geq 0:\ T = T_{\max}(1-At)$$

$$Z = x - x_o - 1.26\left(T_{\max} - T_a\right)^{\frac{1}{3}}\left(\frac{t}{1+2t}\right)$$

eqn. 33

Split into two parts eqn. 33 sets a distance, x_0, beneath the shocked surface where no thermal gradient exists at any one time. This distance is of course a function of the severity of cooling as are the constants A and T' which allow bulk cooling through time and location of the minimum temperature at the end of the cooling cycle.

The excellent correlation is however only valid for the form of air jet shock tests conducted by Hasan. Creating a diverging envelope of localised forced convection along side faces of the specimen as well as the cooled edge creates a scenario more conducive to two dimensional analysis. It is also likely in these tests a thermal gradient was achieved through the specimen thickness producing even a third dimension consideration. As it will be seen these apparently large differences do not produce significantly large changes in the final stress distribution.

5.1.5 – Thermomechanical Stress

5.1.5.1 – TIMOSHENKO'S NET STRESS SOLUTION

A one dimensional stress distribution can be calculated using Timoshenko's thermoelastic consideration [128] given in eqn. 34. Figure 26 shows the typical stress profiles produced by symmetric and non-symmetric thermal shock. The bending effect induced by non-symmetric cooling is determined by the third term of eqn. 34

Non-symmetric or single edge shock

$$\sigma_z = -\alpha E T + \frac{\alpha E}{W}\int_0^W T dx + \frac{12\alpha E}{W^3}\left[x-\frac{W}{2}\right]\int_0^W T\left[x-\frac{W}{2}\right]dx$$

Symmetric or equal double edge shock

$$\sigma_z = -\alpha E T + \frac{\alpha E}{W}\int_0^W T dx$$

eqn. 34

where α is the linear coefficient of thermal expansion and W is the specimen width with zero at the shocked edge for single edge shock.

Figure 27 shows a diagrammatic representation of the superposition of thermal stress effects. Initially a compressive stress is applied to completely suppress the uniform thermal expansion equal to $-\alpha ET$, where T is the temperature of a point. To this is added the sum of stress equivalent to the uniform expansion αET at each point as determined by the thermal distribution. A bending term is then added to account for static imbalance caused by non-symmetric cooling.

It is likely for non-symmetric cooling that two functions similar to that of eqn. 31 would adequately represent cooling over each edge, as illustrated by Figure 28. If this is the case then eqn. 34 can be rewritten as eqn. 35.

$$\sigma_z = -\alpha ET + \frac{\alpha E}{W}\int_0^{x_1} T dx + \frac{\alpha E}{W}\int_{x_1}^{W} T dx$$

$$+\frac{12\alpha E}{W^3}\left[x-\frac{W}{2}\right]\int_0^{x_1} T\left[x-\frac{W}{2}\right]dx + \frac{12\alpha E}{W^3}\left[x-\frac{W}{2}\right]\int_{x_1}^{W} T\left[x-\frac{W}{2}\right]dx$$

where *for* $0 \le x \le x_1$ then $T = T_1 = f_1(x, t)$ and $0 \le x \le x_1$ then $T = T_2 = f_2(x, t)$

eqn. 35

This is purely a thermal stress consideration, should a mechanical load be present to represent body forces or component mechanical working then the summation of same direction stresses allows the Timoshenko equation to be modified into the following.

Non-symmetric or single edge shock with mean mechanical load

$$\sigma_z = -\alpha ET + \frac{\alpha E}{W}\int_0^{W} T dx + \frac{12\alpha E}{W^3}\left[x-\frac{W}{2}\right]\int_0^{W} T\left[x-\frac{W}{2}\right]dx + \sigma_M$$

eqn. 36

Symmetric or equal double edge shock with mean mechanical load

$$\sigma_z = -\alpha ET + \frac{\alpha E}{W}\int_0^{W} T dx + \sigma_M$$

eqn. 37

where σ_M is the applied load perpendicular to one dimensional heat flow.

5.1.6 – Stress Intensity Factors

5.1.6.1 – A GREEN'S FUNCTION APPROACH

An analytical calculation of stress intensity factors for single-edge cracked plates subject to thermal stress is given by Emery *et al.* Having determined the elastic thermal stress distribution, a Green's function for an arbitrary stress distribution can be utilised to calculate the stress intensity factor. This factor applies for crack lengths in the tensile region of the uncracked stress solution only. Given by eqn. 38, σ_T is the thermal stress distribution in the uncracked state and $G(x,a,W)$ is the Green's function accounting for specimen geometry and crack length.

$$K_I = \frac{2}{\sqrt{\pi a}} \int_0^a \sigma_T G\left(\frac{x}{a}, \frac{a}{W}\right) dx$$

eqn. 38

The stress intensity range is given by eqn. 39, where K_{MAX} and K_{MIN} are the maximum and minimum stressed states of the thermal shock cycle respectively.

$$\Delta K = K_{MAX} - K_{MIN}$$

eqn. 39

It is not always the case that the maximum stress state occurs at the end of the shock cycle. Emery stated that stresses decrease with time. However, the tensile stress at the shocked edge can demonstrate a rapid rise and fall throughout the cycle and the point of zero stress closest to the shocked edge moves inward at a decelerating rate. Emery showed that a relationship exists between these two factors causing the time of maximum stress intensity not to be solely linked with the fall of tensile stress at the edge. As time periods decrease and crack lengths are close to the shocked edge the largest gradients in stress across the crack create the greatest intensity at the crack tip. As the crack tip moves towards the zero stress point the reduction in stress across the crack takes place at a lesser rate mirrored in a more gradual change in stress intensity factor through the cycle. Consequently at crack lengths close to the zero point in stress the maximum stress intensity will likely be at the end of a short time period cycle.

This is more simply illustrated in Figure 29. A position x' exists where stresses from different points in time may intersect and consequently stresses behind this point would raise and fall over the shock cycle for large thermal gradients and small time periods. If a crack of length a is smaller than x' then the load of stress on the crack tip

will oscillate in this fashion causing a proportionate oscillation in stress intensity through the cycle, resulting in a maximum stress intensity at some point inside the cycle. Ahead of the point x' tensile stresses only increase but at a decelerating rate. Therefore for crack lengths greater than x' loading stresses would cause stress intensities to mirror a decelerating increase through the applied shock cycle, causing the maximum stress intensity to exist at the end of the cycle. Whether such a position as x' exists for a particular thermal shock load would certainly be a function of its severity. It is likely that initial oscillations in near surface stresses are caused by the rate of heat extraction by forced convection exceeding that of the rate of heat conduction to the surface by the materials own thermal conductivity.

5.2 – Numerical Theory

5.2.1 – Finite Difference Forced Convection Determination

5.2.1.1 – NON-TEMPERATURE DEPENDANT THERMAL PROPERTIES

The energy balance method is used to set-up a set of linear equations which can be used to describe a transient state heat transfer problem. By equating the sum of all heat flow rates to the overall heat flow rate in the material, as in eqn. 40 a straight forward series of algebraic manipulations are required,

$$\dot{Q}_f + \dot{Q}_k = mC_P \frac{\partial T}{\partial t} = \rho V C_P \frac{\partial T}{\partial t}$$

eqn. 40

where Q_f and Q_k are the heat flow rates by convection and conduction determined by Fourier and Newton respectively.

$$\dot{Q}_f = hA(T_S - T_a) \quad \text{and} \quad \dot{Q}_k = kA\left(\frac{\partial T}{\partial x}\right)$$

A diagrammatic representation of a one dimensional model is given in Figure 30 with a perfectly insulated edge and rapidly cooled opposing edge, a direct interpretation of Figure 24.

Two equations are required to fully describe the system, one at the shocked node and another to describe internal nodes. Partial derivatives are estimated in the standard finite difference manner, with a central difference used with respect to distance x and forward difference used with respect to time t. Assuming heat flow is towards the current node and elements are square and equal in size eqn. 40 leads to eqn. 41.

$$\text{Shocked Node:} \quad T_s^{t+1} = 2Fo\left[\left(\frac{1}{2Fo} - 1 - Bi\right)T_s^t + T_{s+1}^t + BiT_a\right]$$

$$\text{Internal Node:} \quad T_i^{t+1} = Fo\left(T_{i-1}^t + T_{i+1}^t\right) + T_i^t\left(1 - 2Fo\right)$$

$$\text{where} \quad Fo = \frac{k\Delta t}{\rho Cp \Delta x^2} \text{ and } \quad Bi = \frac{h\Delta x}{k}$$

eqn. 41

A stability condition is easily obtained from the shocked node equation such that the coefficient to T_s^t remains greater than 0. This premise results in the condition given

by eqn. 42 showing the time increment Δt must not exceed the value otherwise oscillations will occur in the solution.

$$\Delta t \leq \frac{\rho Cp \Delta x^2}{2k(Bi+1)}$$

eqn. 42

The value of h, the surface heat transfer coefficient, is the single value which determines the severity of thermal shock loading for a given material. These definitions can easily be written into a computer programme and used to determine the required h value for a given start and end temperature over a given time period. This value can then be used to load a finite element thermal distribution model for thermal stress evaluation.

A simple linear iteration procedure is used to reduce an initial guess for h to the required value. Using the error between the final end temperature and the required end temperature as a percentage the original h value is reduced or increased for another run. The programme is terminated when the error is within 0.1%. The following shows the straight forward calculation required.

$$T_{factor} = \frac{T_{current} - T_{required}}{T_{required}}$$

$$h_{new} = h_{current} + \left(h_{current} \cdot T_{factor}\right)$$

As well as being straightforward to introduce into code this simple form has the added advantage of never overshooting the required solution.

Convergence is very rapid, completing in around five iterations for an initial error of 50%. However with smaller thermal gradients the number of iterations can be doubled since only small h values are required inducing smaller changes in T_{factor}.

Once completed Timoshenko's equation can be used to determine the resulting stresses orthogonal to the model plane at any recorded time during the downshock cycle. Such a programme models a full edge shock on the plate, in order to determine the effect of localising the shock on the thermal gradient and surface heat transfer coefficient a two dimensional model was generated on the same principles as above. The number of equations required however is increased to ten and are listed below in conjunction with Figure 31 moving left to right and top to bottom.

Top left corner $$T_P^{t+1} = Fo\left[2T_E + 2T_S + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Top side inner $$T_P^{t+1} = Fo\left[T_W + T_E + 2T_S + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Top right corner $$T_P^{t+1} = Fo\left[2T_W + 2T_S + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Left side inner $$T_P^{t+1} = Fo\left[T_N + T_S + 2T_E + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Internal $$T_P^{t+1} = Fo\left[T_N + T_S + T_E + T_W + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Right side inner $$T_P^{t+1} = Fo\left[T_N + T_S + 2T_W + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Shocked region $$T_P^{t+1} = Fo\left[T_N + T_S + 2T_W + 2BiT_a + \left(\frac{1}{Fo} - 2Bi - 4\right)T_P\right]$$

Left symmetry $$T_P^{t+1} = Fo\left[2T_N + 2T_E + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Internal symmetry $$T_P^{t+1} = Fo\left[2T_N + T_E + T_W + \left(\frac{1}{Fo} - 4\right)T_P\right]$$

Shocked symmetry $$T_P^{t+1} = Fo\left[2T_N + 2T_W + 2BiT_a + \left(\frac{1}{Fo} - 2Bi - 4\right)T_P\right]$$

eqn. 43

The most stringent stability condition is present as the T_P coefficient in the shocked node equations and results in the condition of eqn. 44.

$$\Delta t \leq \frac{\rho Cp \Delta y^2}{2k(Bi+2)}$$

eqn. 44

This produces the requirement of smaller time increments to be taken in the solution and coupled with an increase in nodes by at least an order, the number of calculations can enter the millions as opposed to the tens of thousands for a one dimensional model. Run times are consequently increased to fractions of an hour instead of minutes.

There is little or no effect on the thermal distribution at the symmetry plane when including a 2D consideration for localised shock. However the surface heat transfer coefficient is significantly changed depending on the size of the localised

region. Finite element models using this localised shock demonstrate a large reduction in both tensile and compressive stress levels with the generation of a significant lateral compressive stress region not present in full edge shock.

5.2.1.2 – TEMPERATURE DEPENDANT THERMAL PROPERTIES

A re-evaluation of the finite difference equations was made to account for changing thermal properties through the shock cycle. Changes in thermal conductivity and density were accounted for whilst specific heat capacity is considered relatively constant for metals above 0°C.

Again the energy balance method was used, assuming heat flow is towards the current node and elements are square and equal in size. Each node was assigned a separate thermal conductivity and density resulting in separate Fourier terms. For a one dimensional consideration the equation at the shocked node was unchanged. However for internal nodes two Fourier numbers resulted from the two adjacent nodes as eqn. 45.

$$\text{Shocked Node:} \qquad T_s^{t+1} = 2Fo\left[\left(\frac{1}{2Fo} - 1 - Bi\right)T_s^t + T_{s+1}^t + BiT_a\right]$$

$$\text{Internal Node:} \qquad T_i^{t+1} = Fo_1 T_{i-1}^t + Fo_2 T_{i+1}^t + (1 - Fo_1 - Fo_2)T_i^t$$

$$\text{where} \qquad Fo_1 = \frac{k_{i-1}\Delta t}{\rho_i Cp \Delta x^2} \qquad \text{and} \qquad Fo_2 = \frac{k_{i+1}\Delta t}{\rho_i Cp \Delta x^2}$$

eqn. 45

A single density value at the current temperature need only be used since it is only present in equating total heat transfer. However, the following linear equations, eqn. 46, are provided in the solution to describe the thermal conductivity and density for any temperature in the specified range.

$$k = 15 + 0.013T \qquad \text{for } 100°C \leq T \leq 500°C$$

$$\rho = 7900 - 0.5T \qquad \text{for } -196°C \leq T \leq +1000°C$$

eqn. 46

Temperatures encountered in the models will exceed that of the 500°C quoted here. At this point thermal conductivity will still be evaluated according to this criterion. Large deviations in linearity are not expected in the immediately higher range as well as thermal conductivity changing little over the given range.

For stability the most stringent condition is as before, eqn. 42. For a coarse mesh where changes in thermal conductivity may be large, a second stability condition arising from the coefficient of internal node equations may need to be checked. A further condition can be made to ensure permanent stability for cooling models; maximum temperature properties can be used to calculate a maximum usable time increment, Δt. Since it is known for AISI 316 density is inversely proportional and thermal conductivity is directly proportional to increasing temperature, using properties at the maximum temperature of the model will generate the smallest possible time increment and so permanent stability.

However, if heating is taking place the maximum temperature may not be known and consequently an over estimation must be made for the maximum time increment and constantly checked and re-run for stability to be assured.

The two dimensional consideration was made for temperature dependence resulting in the equations listed below again in conjunction with Figure 31.

Top left corner $$T_P^{t+1} = 2Fo_E T_E + 2Fo_S T_S + (1 - 2Fo_E - 2Fo_S)T_P$$

Top side inner

$$T_P^{t+1} = Fo_W T_W + Fo_E T_E + 2Fo_S T_S + (1 - Fo_W - Fo_E - 2Fo_S)T_P$$

Top right corner $$T_P^{t+1} = 2Fo_W T_W + 2Fo_S T_S + (1 - 2Fo_W - 2Fo_S)T_P$$

Left side inner $$T_P^{t+1} = Fo_N T_N + Fo_S T_S + 2Fo_E T_E + (1 - Fo_N - Fo_S - 2Fo_E)T_P$$

Internal

$$T_P^{t+1} = Fo_N T_N + Fo_S T_S + Fo_E T_E + Fo_W T_W + (1 - Fo_N - Fo_S - Fo_E - Fo_W)T_P$$

Right side inner

$$T_P^{t+1} = Fo_N T_N + Fo_S T_S + 2Fo_W T_W + (1 - Fo_N - Fo_S - 2Fo_W)T_P$$

Shocked Region

$$T_P^{t+1} = Fo_N T_N + Fo_S T_S + 2Fo_W T_W + 2CT_{a+}(1 - Fo_N - Fo_S - 2Fo_W - 2C)T_P$$

Left symmetry $$T_P^{t+1} = 2Fo_N T_N + 2Fo_E T_E + (1 - 2Fo_N - 2Fo_E)T_P$$

Internal symmetry $T_P^{t+1} = 2Fo_N T_N + Fo_E T_E + Fo_W T_W + (1 - 2Fo_N - Fo_E - Fo_W)T_P$

Shocked symmetry $T_P^{t+1} = 2Fo_N T_N + 2Fo_W T_W + 2CT_{a+}(1 - 2Fo_N - 2Fo_W - 2C)T_P$

$$\text{where} \quad C = \frac{h\Delta t}{\Delta x \rho C_P} \quad \text{and} \quad \Delta t \leq \frac{\Delta y \rho C_P}{\frac{4k_{max}}{\Delta y} + 2h}$$

eqn. 47

A summation of thermal conductivity values from all adjacent nodes is required for a stability condition, but is here replaced with the maximum temperature conductivity for permanent stability. As well as the change in time incrementation, a further difference to the fixed thermal property solution is that a separate Fourier number is required for each adjacent node, the current node being a function of all these Fourier numbers.

5.2.2 – Finite Element Elastic Thermal Stresses

5.2.2.1 – ELASTICITY

The theory of elasticity is modified by thermal strains to determine expressions of two dimensional plane stress in conjunction with a thermal distribution. Mechanical elastic stresses and strains for a two dimensional element are given by eqn. 48

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + f_x = 0$$

$$\frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{xy}}{\partial x} + f_y = 0$$

$$\varepsilon_x = \frac{\partial u}{\partial x} \qquad \varepsilon_y = \frac{\partial v}{\partial y} \qquad \varepsilon_{xy} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \qquad \varepsilon_y = \frac{\partial w}{\partial z}$$

eqn. 48

and can be rewritten in matrix form as eqn. 49.

$$\begin{bmatrix} \partial/\partial x & 0 & \partial/\partial y \\ 0 & \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{Bmatrix} + \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} = 0 = \mathbf{S}^{\mathrm{T}}\boldsymbol{\sigma} + \mathbf{B} = \mathbf{0}$$

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \end{Bmatrix} = \begin{bmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} = \mathbf{SU}$$

eqn. 49

For a plane stress element, the relationship between stresses and strains can be described by eqn. 50 and manipulated into single stress terms to produce eqn. 51

$$\varepsilon_x = \frac{\sigma_{xx}}{E} - \frac{\nu}{E}\sigma_{yy} \qquad \varepsilon_y = \frac{\sigma_{yy}}{E} - \frac{\nu}{E}\sigma_{xx} \qquad \varepsilon_z = -\frac{\nu}{E}\left(\sigma_{xx} + \sigma_{yy}\right)$$

eqn. 50

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \cdot \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \end{Bmatrix}$$

$$\boldsymbol{\sigma} = \mathbf{D} \cdot \boldsymbol{\varepsilon}$$

eqn. 51

A consideration of plane strain requires a little more manipulation and produces a different ***D*** term. This is purely a mechanical relationship and must therefore be modified to account for thermal strains. Total strain is considered as a summation of both mechanical and thermal strains and so mechanical strain, ε_M, is introduced as a description of total and thermal strain, ε_θ,

$$\varepsilon_T = \varepsilon_M + \varepsilon_\theta$$

$$\boldsymbol{\sigma} = \mathbf{D} \cdot (\boldsymbol{\varepsilon}_T - \boldsymbol{\varepsilon}_\theta)$$

eqn. 52

Once a thermal distribution is known thermal strains can be considered simply as $\alpha\{\Delta T\}$ allowing the relationship of stress to total strain to be written in conjunction with eqn. 49 as eqn. 53.

$$\boldsymbol{\sigma} = \mathbf{D} \cdot (\boldsymbol{\varepsilon}_T - \alpha\Delta T)$$

$$\boldsymbol{\sigma} = \mathbf{D} \cdot (\mathbf{SU} - \alpha\Delta T)$$

eqn. 53

An approximation for the unknown displacement function **U** is made by the finite element method. The modified Galerkin method demonstrates that a continuous problem domain can be represented by a discretised summation of linear terms through that region given by eqn. 54.

$$U'(x,y) = \sum_{i=1}^{n} u_i \phi(x,y) \qquad V'(x,y) = \sum_{i=1}^{n} v_i \phi(x,y)$$

$$\mathbf{U} = \mathbf{\Phi} \cdot \boldsymbol{a}$$

eqn. 54

Approximate x and y displacements at the nodes are represented by u and v respectively and shape functions ϕ describe their function between nodes. Strains are calculated from these functions by eqn. 48 to give eqn. 55.

$$\boldsymbol{\varepsilon} = \mathbf{S} \cdot \boldsymbol{\phi a} = \mathbf{B} \cdot \boldsymbol{a}$$

where $\qquad \boldsymbol{a}^{\mathrm{T}} = \{u_1, v_1, u_2, v_2, u_3, v_3 \ldots u_n, v_n\}$

eqn. 55

5.2.2.2 – GENERAL WORK THEORY

All finite element models have been conducted using the general work theory equating work done by external forces to internal strain energy and can be written as eqn. 56.

$$\int_V \delta\varepsilon\sigma \; dV = \delta a^T f + \int_V \delta U^T b \; dV + \int_V \delta U^T t \; dV$$

eqn. 56

This accounts for applied concentrated loads, body forces and any applied traction loads. Using eqn. 53 and eqn. 55 a description of the finite element solution is as eqn. 57.

$$\int_V B^T DBa \; dV = f + \int_V \Phi^T b \; dV + \int_V \Phi^T t \; dV + \int_V B^T D\alpha[\Delta T] \; dV$$

$$[K]^e \{a\}^e = \{f\}^e$$

$$[K]^e = \int_{Ve} B^T DBa^e \; dV$$

$$\{f\}^e = \{f_p\} + \{f_b\} + \{f_t\} + \{f_\theta\}$$

eqn. 57

The advantage of this method is the summation of required effects. The user is able to specify the form of loading as is required, without accounting for any unnecessary parameters. It is also very convenient to load a model under its constitutive conditions to obtain component solutions of a superposition model. For example, a pressure vessel under combined direct and bending stress could be

analysed separately for the direct stress and bending stress solutions. Providing no large thermal stresses are present and the material is elastic the two separate solutions can be determined for independent study or combined together for an overall response. The disadvantage is the solution can easily be an inadequate description of the problem should a subtle aspect to the problem be neglected, for example, body loads can be relevant in large structures as a description of a components mass.

5.2.3 – Numerical Fracture Mechanics

5.2.3.1 – QUARTER-POINT METHOD

The complexities of determining a valid crack tip descriptor can to some extent be eased by a numerical approach. Here the finite element quarter-point method is utilised to generate a singularity effect. By positioning midside nodes of quadrilateral second order elements to a quarter distance from a corner node, an inverse square relationship on distance from that corner node is achieved. For a one dimensional element Henshell and Shaw show strain to be described by an inverse square root as eqn. 58.

$$\varepsilon = \frac{dU}{dr} = u_1\left(1 - \frac{3}{2\sqrt{2r}}\right) - u_2\left(2 - \frac{2}{\sqrt{2r}}\right) + u_3\left(1 - \frac{1}{2\sqrt{2r}}\right)$$

eqn. 58

Figure 32 shows the mesh structure originally used and the contemporary method used in the current work. Degeneration of elements surrounding the crack tip by collapsing two nodes together at the singularity to form a degenerated triangle from a quadrilateral provides a slight improvement in evaluation of a contour integral, though care must be taken not to induce derogatory element distortions. Mesh studies made in the work on this structure of model have shown that quoted contour integrals remain constant for a variety of mesh densities, whilst only the highest densities generate appropriately refined crack tip stresses. However, near tip stress refinement can be achieved sufficiently well with a moderate density.

A measure of energy release rate with crack advancement, a contour integral evaluation in finite element is given by ABAQUS [129] as eqn. 59.

$$\overline{J} = \int_s \lambda(s) nHq \; ds$$

$$\text{where} \qquad H = \left(WI - \sigma \frac{\partial u}{\partial x} \right)$$

eqn. 59

For an elastic material W is the elastic strain energy. The remaining terms, $\lambda(s)$, n, q and s are virtual crack advance, the outward normal to s, the virtual crack propagation direction and the surface contour surrounding the crack tip respectively.

Using this method stress intensity values can be determined directly or indirectly from J-Integral values and cracked edge displacements respectively. For plane stress, the theoretical relationship between J and K is as eqn. 60

$$J = \frac{K_I^2 + K_{II}^2}{E} \quad \Rightarrow \quad K_I = \sqrt{JE} \qquad \text{contour method } (K_{II} = 0)$$

eqn. 60

and a numerical method determined from near tip displacements for a symmetrically loaded crack is given as eqn. 61

$$K_I = \frac{2\mu\sqrt{\pi}}{1 + \left(\frac{3-\upsilon}{1+\upsilon} \right)} \lim_{r \to 0} \frac{2u_y}{\sqrt{2r}} \qquad \text{direct method}$$

$$\mu = \frac{E}{2(1+\upsilon)}$$

eqn. 61

where μ is the shear modulus and u_y refers to the y-displacement of a cracked edge node approaching the crack tip.

It is important at this point to note constraints placed on the crack tip nodes. Many coincident nodes are generated at the crack tip as a result of the modelling process and are consequently able to fan out on loading. Though this may have advantages in possible applications to plastic flow and crack tip blunting near tip displacements were found to be erratic and unpredictable. Any application to crack tip blunting would also be partially dependant on the number of coincident nodes.

By placing a dependant constraint on these nodes this erratic behaviour was eliminated. All coincident nodes except one were grouped together and displacements constrained to equal that of the remaining node which was left to displace accordingly. Such a constraint may be written as eqn. 62.

$$U_X^{TIP-n} = U_X^n$$

eqn. 62

This demonstrates how the *x*-direction displacements of a set of nodes at the crack tip, *TIP-n*, are equated to that of the displacements of a single node, *n*. This remaining node was chosen as the crack tip node of the element ending the crack face. This allowed the flow of displacements at the cracked edge to perform as a continuous face, i.e. displacements at the crack tip node were driven by the opening of the crack immediately before it, as opposed to coincident nodes or nodes ahead of the crack tip.

5.2.4 – Model Verification

5.2.4.1 – MODEL SCOPE

Elastic finite element test models were conducted at 5mm crack lengths, with a load of 75MPa on two separate types of model;

- Centre cracked thin square plate
- Thin edge-cracked rectangular plate.

These models allow a plane stress evaluation to be made of the required geometries, types of deformation and their behaviour on the singularity inclusion. No thermal verification is made in regard to singularity behaviour as it is felt only non-parallel heat flow would require study. Since this study is based on parallel heat flow, verification is applied to mechanical behaviours only.

5.2.4.1 – VERIFICATION METHODOLOGY

To determine the validity of numerical fracture mechanics two classical uniaxial scenarios were conducted and compared to established theory. By modelling a series of crack lengths, and evaluating them by stress intensity factor and plastic zone, with and without a dependant constraint at the crack tip nodes, the applicability to LEFM can be determined.

Verification of a uniaxially loaded centre cracked square specimen and edge-cracked rectangular plate with unrestrained bending was conducted to mirror the geometric constraints of biaxial and thermal shock models. Figure 33a shows the basic centre cracked square geometry with symmetry constraints whilst Figure 33b shows the rectangular edge cracked construct (x marks locations of fixed constraints). A

single transverse constraint is placed at the centre of the top face of scenario *b* to allow elongation with applied loads and free rotation to allow bending, as would also be present in non-symmetric thermal shock. Both scenarios are given by the classical solutions of eqn. 63 below as provided by Rooke and Cartwright.

$$\frac{K_I}{K_O} = \frac{1 - 0.5\left(\frac{a}{b}\right) + 0.326\left(\frac{a}{b}\right)^2}{\sqrt{1 - \left(\frac{a}{b}\right)}} \qquad \text{Centre cracked plate}$$

$$\frac{K_I}{K_O} = 1.12 - 0.23\left(\frac{a}{b}\right) + 10.6\left(\frac{a}{b}\right)^2 - 21.7\left(\frac{a}{b}\right)^3 + 30.4\left(\frac{a}{b}\right)^4 \qquad \text{Edge cracked plate}$$

$$K_O = \sigma\sqrt{\pi a}$$

eqn. 63

The centre crack solution was obtained by Isida with boundary collocation of complex stress functions for large *h/b* values (≥1) and all *a/b* values. A solution for the edge-cracked model was obtained through a collocation method by Brown and Srawley in the range of $h/b \geq 1.0$ and $a/b \leq 0.6$. The term K_O is the stress intensity factor for an isolated crack of length $2a$ in a plate subjected to a uniaxial load σ.

Verification of stress intensity values are made by eqn. 60, eqn. 61 and eqn. 63. Plastic zone sizes and shapes are also verified based on the biaxial LEFM solution described in the following chapter (section 6.1). At an angle of zero degrees and a uniaxial load the solution can be greatly simplified to eqn. 64.

$$r_{PY} = \frac{K_I^2}{2\pi\sigma_{yield}^2}$$

$$r_{PX} = \frac{K_I^2}{2\pi\left(\sigma_{yield} + \sigma_T\right)^2}$$

eqn. 64

This will provide the magnitude of the plastic zone size at an angle of zero degrees ahead of the crack tip. Its derivation is discussed in the following sections.

5.3 – Fracture Analysis

5.3.1 – Biaxial T-Stress

5.3.1.1 – COMPONENT STRESSES

Biaxial loading is the next logical step from uniaxial loading in assessing the structural integrity of components. It is not inherently related to mixed mode loading, however the two can easily occur together. The effects of biaxiality are quantified as a non-singular term and referred to as the T-stress, σ_T, affecting the lateral stress distribution ahead of the crack tip but not the stress intensity factors. This does not however stop the σ_T from affecting crack propagation. The σ_T is included into the description of component stresses ahead of a crack tip by accounting for the higher order terms of a Taylor series expansion in the original biaxial derivation of the stress intensity factor. Here it is quoted in eqn. 65 as determined by Hua, taking the form of a non-singular term added to the lateral stress component.

$$\sigma_y = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1+\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]+\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\cos\frac{\theta}{2}\cos\frac{3\theta}{2}$$

$$\sigma_x = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]-\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left[2+\cos\frac{\theta}{2}\cos\frac{3\theta}{2}\right] + \sigma_N[1-\Lambda]\cos(2\alpha)$$

$$\sigma_{xy} = \frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\sin\frac{\theta}{2}\cos\frac{3\theta}{2}+\frac{K_{II}}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]$$

eqn. 65

The state of stress near the crack tip consequently becomes a function of the stress intensity factor and the σ_T. The σ_T being defined by the stresses acting normal and lateral to the crack tip and the angle, α, of the crack to an applied primary load. The parameter Λ is the ratio of biaxiality around the crack tip.

$$\Lambda = \frac{\sigma_Q}{\sigma_P}$$

Where σ_P and σ_Q are the normal and lateral stresses acting on the crack respectively. Kfouri and Miller described the normal, σ_N, and lateral, σ_L, applied loads in terms of σ_P and σ_Q as in eqn. 66;

$$\sigma_N = \frac{1}{2}(\sigma_P + \sigma_Q) + \frac{1}{2}(\sigma_P - \sigma_Q)\cos 2\alpha$$

$$\sigma_L = \frac{1}{2}(\sigma_P + \sigma_Q) - \frac{1}{2}(\sigma_P - \sigma_Q)\cos 2\alpha$$

eqn. 66

Solving simultaneously for $[\sigma_L - \sigma_N]$, σ_Q and σ_P can be described as eqn. 67

$$\sigma_Q = \sigma_N + \frac{\sigma_L - \sigma_N}{2}\left[1 + \frac{1}{\cos 2\alpha}\right]$$

$$\sigma_P = \sigma_N + \frac{\sigma_L - \sigma_N}{2}\left[1 - \frac{1}{\cos 2\alpha}\right]$$

eqn. 67

Since σ_N and σ_L are known as applied loads distant from the crack, a full description of the near tip stress state can now be determined.

5.3.2 – Biaxial Plastic Zone

Kfouri and Miller, to quantify the state of applied biaxial loading, have used a parameter $[\sigma_L - \sigma_N]$. The value Λ has also been used similarly by Hua. All these values however, must be used in conjunction with the stress intensity factor leading to a cumbersome dual parameter approach. It therefore seems natural to consider the plastic zone, for, whilst small in relation to the crack length, the plastic zone is a direct consequence of the crack tip state. Hua incorporated the σ_T into a mixed-mode von Mises plastic zone with good results. However, only the plastic zone radius ahead of the crack tip, r_P^*, was used. It is possible, however, for this parameter to be inadequate as the same r_P^* value can be created by different combinations of K_I and σ_T. The result being the same r_P^* value for two different von Mises plastic zone shapes.

To overcome this a parameter describing the shape and size of the von Mises plastic zone is proposed.

5.3.2.1 – BIAXIAL PLASTIC ZONE FUNCTION

For a linear-elastic material under mixed mode loading but not considering σ_T the von Mises plastic zone magnitude is a function of K_I and K_{II} in the from of eqn. 68.

$$r_{PvM} = f(K_I^2, K_{II}^2, K_I K_{II})$$

eqn. 68

This means all loading conditions can be described by the stress intensity factor, and consequently the shape of the plastic zone remains constant whilst its size is variable. However, when T-Stress is included the relationship becomes more complex, being related to K_I and K_{II} and σ_T by eqn. 69.

$$r_{PvM} = f(K_I^2, K_{II}^2, K_I K_{II}, K_I, K_{II}, \sigma_T)$$

eqn. 69

Since σ_T is effective only on the lateral stress component, σ_x, the von Mises shape will be similarly affected; should the lateral stress component be increased the lateral plastic zone will extend and be reflected in the von Mises distribution. Consequently the shape of the von Mises plastic zone is controlled by σ_T and can therefore no longer be considered constant.

For single mode loading the plastic zone is permanently symmetrical about the crack plane. It is therefore proposed that the radius ahead of the crack tip and the normal height uniquely described the shape of the plastic zone, i.e. the length and height of the von Mises plastic zone. The ratio of these two dimensions should therefore uniquely express the shape of the von Mises plastic zone. For oscillating loads, the change in these dimensions can be used to account for stress range. Such a ratio is given by eqn. 70.

$$\textit{von Mises plastic zone shape ratio} = \frac{\Delta r_{PvML}}{\Delta r_{PvMH}}$$

eqn. 70

In order that magnitude is accounted for, the absolute size of the plastic zone must be introduced. By using the length of the plastic zone created by stress normal to the crack plane, σ_y for non-inclined cracks, the primary normal load is accounted for. This stress component is advantageous since it is the primary stress acting on the crack and will not be affected by any changes in σ_T. It is proposed that a parameter

providing a unique description of the plastic zone is given by eqn. 71 and illustrated in Figure 34.

$$\frac{\Delta r_{PvML}}{\Delta r_{PvMH}} \cdot \Delta r_{Py}$$

eqn. 71

A theoretical determination of this function is required to test the assumption that the shape ratio is a unique description of biaxial loading on a crack tip. This determination follows in the next section, and though mixed mode loading is not considered in the final analysis, it has been included for completeness and for further work.

5.3.2.2 – THEORETICAL MIXED-MODE-BIAXIAL VON MISES PLASTIC ZONE

Using eqn. 65 and the von Mises yield criterion of eqn. 72, plane stress and plane strain functions can be derived for a mixed mode biaxial von Mises plastic zone and are both presented in eqn. 73 and eqn. 74.

$$2\sigma_{yield}^2 = (\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2$$

$$\sigma_{1(+),2(-)} = \frac{\sigma_x - \sigma_y}{2} \pm \sqrt{\left[\frac{\sigma_x - \sigma_y}{2}\right]^2 + \sigma_{xy}^2}$$

$$\sigma_3 = 0 \qquad \textit{for plane stress}$$

$$\sigma_3 = \nu(\sigma_1 + \sigma_2) \qquad \textit{for plane strain}$$

eqn. 72

Appendix A shows the derivation in more detail, the final solution for both plane strain and plane stress conditions is shown below.

For plane stress: -

$$\sigma_{yield}^2 = \frac{K_I^2}{2\pi r}\cos^2\frac{\theta}{2}\left[3\sin^2\frac{\theta}{2}+1\right]+\frac{K_I K_{II}}{2\pi r}\sin\theta\left[3\cos\theta-1\right]$$
$$+\frac{K_{II}^2}{2\pi r}\left[3+\sin^2\frac{\theta}{2}\left[1-9\cos^2\frac{\theta}{2}\right]\right]$$
$$+\sigma_T\left[\frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-3\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]\right]$$
$$-\sigma_T\left[\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left[4+3\cos\frac{\theta}{2}\cos\frac{3\theta}{2}\right]\right]$$
$$+\sigma_T^2$$

eqn. 73

For plane strain: -

$$\sigma_{yield}^2 = \frac{K_I^2}{2\pi r}\cos^2\frac{\theta}{2}\left[3\sin^2\frac{\theta}{2}+(1-2\nu)^2\right]+\frac{K_I K_{II}}{2\pi r}\sin\theta\left[3\cos\theta-(1-2\nu)^2\right]$$
$$+\frac{K_{II}^2}{2\pi r}\left[3+\sin^2\frac{\theta}{2}\left[(1-2\nu)^2-9\cos^2\frac{\theta}{2}\right]\right]$$
$$+\sigma_T\left[\frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[(1-2\nu)^2-3\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]\right]$$
$$-\sigma_T\left[\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left[(1-2\nu)^2+3\cos\frac{\theta}{2}\cos\frac{3\theta}{2}+3\right]\right]$$
$$+\sigma_T^2\left(1-\nu+\nu^2\right)$$

$$\text{where} \qquad \sigma_T = \sigma_N(\Lambda-1)\cos 2\alpha$$

eqn. 74

The von Mises plastic zone radius, r, of eqn. 73 and eqn. 74 must consequently be obtained by solving as a quadratic for $1/\sqrt{r}$. This is more clearly shown if eqn. 73 and eqn. 74 are reduced to non-inclined cracks with $\theta = 0$.

For plane stress: -

$$\sigma_{yield}^2 = \frac{K_I^2}{2\pi r}+\frac{\sigma_T K_I}{\sqrt{2\pi r}}+\sigma_T^2$$

For plane strain: -

$$\sigma_{yield}^2 = \frac{K_I^2(1-2\nu^2)}{2\pi r}+\frac{\sigma_T K_I(1-2\nu^2)}{\sqrt{2\pi r}}+\sigma_T^2(1-\nu+\nu^2)$$

eqn. 75

In general the quadratic can be described as

$$\frac{a}{r}+\frac{b}{\sqrt{r}}+c=0$$

eqn. 76

Once the quadratic solution is made it is not immediately clear which of the two resulting solutions is correct. Both plastic zone shapes can be similar under biaxial and mixed mode loading and are equal when $\sigma_T = 0$. By substituting both obtained values of r back into the quadratic of eqn. 73 or eqn. 74 only the positive version of the quadratic makes a solution under biaxial loading. In fact the negative solution of the quadratic performs the opposite biaxial effect on the plastic zone.

A key term within the quadratic solution is σ_T^2 - σ_{yield}^2, which occurs as part of the denominator when solving for r. Under the conditions of $\sigma_T = \sigma_{yield}$ a division by zero occurs for the correct positive solution. Consequently the linear-elastic conditions of $-\sigma_{yield} < \sigma_T < +\sigma_{yield}$ must be imparted to the initial problem.

5.3.3 – Summary of Stress Intensity Determination

With contemporary finite element methods, practical determination of this plastic zone parameter is relatively easy. Previous mathematically intensive methods to determine stress intensity factors can be replaced with very accurate numerical approximations. Either through direct modelling of the crack tip with quarter point elements or by calculating changes in strain energy through release of multi-point constraints. In this work only the quarter point element technique was used. For verification models this method provided accurate determination of stress intensity factors to within less than 1% and 10% for plastic zones sizes.

Using the quarter-point method excellent plastic zone shapes for component stresses and von Mises stresses can be determined. However, using the more straightforward meshing capabilities of multi-point constraints the plastic zones are not as accurate. Though, once built, a change in strain energy analysis can determine stress intensity factors to be used in the above formulation.

5.4 – Quantification of Thermal Loading

5.4.1 – Overview

Past studies of thermal shock loading [48,106-110,116] have successfully used the dimensionless Biot number, β, to describe the thermal downshock condition and is given in eqn. 77.

$$\beta = \frac{h \cdot L}{k}$$

eqn. 77

Essentially this number represents the magnitude of transient loading, h, and thermal resistance provided by the material and geometry, L/k. These past studies have been fuelled on understanding the overall nature of thermal shock for a wide variety of Biot numbers. The time scale, t, over which these shocks are applied are normally given in non-dimensional time, t', or Fourier number, Fo, defined in eqn. 78.

$$t' = \frac{k \cdot t}{\rho \cdot C_P \cdot L^2} = Fo$$

eqn. 78

For thermal down-shock loading both eqn. 77 and eqn. 78 are required, however h is an empirical value based on thermal and fluidic characteristics of the component and film medium respectively. Correlating methods are available such as Nusselt numbers, however, a numerical analysis is preferred since this allows greater freedom and accuracy in the determination of heat transfer coefficients. By providing an explicit description of the thermal downshock, i.e. start, end and ambient temperatures and a time period over which these occur the definition of loading is invariably more descriptive but difficult to use in relation to any further analysis. It would therefore be convenient if an empirical description of thermal down-shock could be simply related to the resulting stress state using only the explicit definition values. Relating this to a description of bounding isothermal crack growth rates would allow final description of the thermal shock crack growth rates by explicit definition of thermal loading.

This would require a description of biaxiality. It is proposed to determine an estimation of the non-singular biaxial stress, σ_T, under isothermal conditions by the stresses induced from thermal downshock loading. In order to do this the stresses must

be related to the surface heat transfer coefficient, h and is addressed in the following section.

5.4.2 – Thermal Loading Evaluation

As described previously the thermal loading is to be defined by a time period and the initial, end and ambient temperatures. The severity of thermal loading is initially described by the normalised thermal gradient in relation to ambient conditions.

$$T_{FACTOR} = \frac{T_{END} - T_A}{T_{INITIAL} - T_A}$$

eqn. 79

Time period is an independent effect and must therefore be included as a separate variable in a function describing h. The form of such a function is proposed to be a power law as in eqn. 80.

$$h = f\left(T_{FACTOR}^{n}, t_P\right)$$

eqn. 80

This is reasoned on the simple basis of exponential cooling. As T_{END} approaches T_A the lower the natural cooling rate and consequently the higher h must be to force the thermal distribution to T_{END}.

5.4.3 – Thermoelastic Stress

The maximum stress calculated at the shocked edge through the cycle as opposed to the stress at the end of the cycle is a more characteristic and relevant measure of loading. When determining a stress intensity factor range, for instance, the maximum and minimum stress states are required to fully describe the extremes of loading on the crack tip. Therefore it is necessary that the time of maximum stress is used as opposed to the stress at the end of the cycle. Hence, it is more likely that a good relationship to the h value can be determined during this period of transient behaviour.

The stress at the shocked surface, σ_{MAX}, is used to characterise the overall stress state through the model. Since the profile of thermal shock stress is predictable, any

point on the profile could be used. However, the most meaningful to fracture mechanics and hence to crack growth rates is that at the shocked surface, since it is this stress which governs the initial propagation of edge cracks.

$$\sigma_{MAX} = f(h, E\alpha)$$

eqn. 81

Therefore the maximum elastic thermal shock stress produced by loading described by T_{FACTOR} of eqn. 80, must be said to be described in the form of eqn. 81.

In order for this thermal stress, σ_{MAX}, to be related to an isothermal biaxial stress, σ_T, a relation must be determined. The relationship is likely to be one related to the biaxiality present in thermal shock and the crack geometry, centre cracked or edge cracked for instance. Such a relationship can only partially be determined analytically and it is inevitable that empirical evidence must be used to determine the exact nature of the relationship described below in eqn. 82.

$$\sigma_T = f(B\sigma_{MAX})$$

eqn. 82

The constant B can only be found by further long term experimental study, however, the experimental evidence [124] presented here in later sections provides a good starting point and is elaborated on in further sections. As such the final step in the proposed correlation of thermal shock to isothermal crack growth rates is proposed in eqn. 83.

$$\frac{da}{dN} = f\left(C \cdot \left[1 - \frac{\sigma_T}{\sigma_N}\right](\Delta K_I)^m\right)$$

eqn. 83

The form of this function is essentially equivalent to the findings of Brown and Miller [31] where increasing negative σ_T was found to accelerate crack growth. In eqn. 83 when σ_T becomes more negative in relation to some positive nominal stress, such as the mean stress of a sinusoidal cycle, crack growth rates are increased.

6 – Results

6.1 – Model Verification

Elastic finite element test models were conducted at a 5mm crack length with a load of 75MPa on two separate types of model: a centre-cracked thin square plate (CCP) and a thin edge-cracked rectangular plate (ECP). This allows a plane stress evaluation to be made of the required model geometry, type of deformation and their behaviour on the singularity. No thermal verification was made in regard to the singularity as it is felt only non-parallel heat flow would require study. Since this study is based on parallel heat flow, verification is applied to mechanical behaviours only.

Table 14 through Table 18 present results of the models conducted with a variety of mesh refinements. Table 14 and Table 16/Table 17 show the CCP and ECP studies respectively, whilst Table 15 and Table 18 show the respective classical solutions.

Classical stress intensity factors are determined by eqn. 63, with plastic zones from eqn. 64. Both compare extremely well maintaining an excellent relation to the classical solution. Though component plastic zone sizes are seen to agree very well with classical size in the CCP the relation is not as good for that of the ECP model. This is likely due to the less straightforward manner of boundary and loading conditions. Where uniaxial load will apply a relatively predictable stress at the crack tip region, any bending taking place could easily differ from the classical solution. In this case boundary conditions both parallel and perpendicular to the crack plane are required in the finite element model. Symmetry conditions are straight forward, whilst the entire top edge of the rectangular model parallel to the crack plane and perpendicular to the applied load is constrained from transverse motion. Consequently, bending is partially constrained and it is likely that this stiffer condition produces higher stresses around the crack tip, and consequently larger plastic zone sizes.

The classical edge-cracked solution as determined by eqn. 63 is applicable for all height to width ratios (h/b) greater than one. Therefore, finite element models requiring boundary conditions both parallel and perpendicular to the crack plane solution should demonstrate independence of parallel constraint. Two types of constraint were conducted; a pin-joint at the centre of the top edge allowing rotation

and free deformation, and a complete lateral constraint indicative of a built-in top-edge presented in Table 16 and Table 17 respectively. The results in plastic zone are essentially equal. This supports the behaviour of the finite element model though it demonstrates its conservative estimation of plastic zone size.

6.2 – Thermal Analysis Results

6.2.1 – Finite Difference Model Refinement

Refinement of the explicit finite difference model was conducted using the surface heat transfer coefficient, thermal distribution, surface cooling and initial cooling gradient as indicators of a concurrent solution. Both time increment and element density are evaluated in a fixed property one dimensional model, this is later extended into temperature dependant (appendix B.1).

A series of models were conducted shown in Table 19 through 21 for 3 separate levels of thermal downshock with properties specified at the maximum temperature. Maximum run times of the models were 6, 12 and 13 minutes converging in 5, 8 and 10 iterations for Table 19 through 21 respectively. Results of these models are shown in Figure 35 through 46 and are discussed in the following sections.

6.2.1.1 – SURFACE HEAT TRANSFER REFINEMENT

Refinement of the surface heat transfer coefficient is shown in Figure 35 to Figure 37 for the three separate levels of downshock. Both stable and unstable regions in time incrementation are shown, and demonstrate the occasional capacity of the model to run satisfactorily outside stable conditions. In most cases however, unstable time incrementation generates large divergences and accelerates the solution wildly out of control and unable to complete. In coarse meshes, the surface heat transfer coefficient cannot be refined even with the smallest of time incrementation, even though values may become concurrent. Increasing element density causes rapid, then permanent refinement of the solution in space and in any region of stable time incrementation.

In comparison to a fixed property one dimensional theoretical value determined from the eigenvalue solution, the finite difference solution compares within 10% of refined values.

6.2.1.2 – THERMAL DISTRIBUTION

Using refined time incrementation values for the three downshock levels, the effect of element density on thermal distribution is evaluated in Figure 38 to Figure 40. It can be seen that a very rapid refinement is achieved in the parabolic region of the distribution requiring only a few elements. Refinement of nodal position values is clearly excellent, showing a general trend of large refinement error only at the intermediate positions in highly coarse meshes. Theoretical values determined from the eigenvalue solution are also plotted and demonstrate approximately a 2% vertical deviation of the finite difference model from the theoretical.

6.2.1.3 – COOLING CURVE

Cooling rates of the shocked surface with refined time incrementation are presented in Figure 41 to Figure 43 with theoretical values shown for comparison. A large refinement error is present in coarse meshes as would be expected from the reduced level of heat flow to the surface from bulk material. Refinement is reached at 50 elements requiring time incrementation in the order of hundredths of a second. However, the initial cooling gradient appears unable to refine producing an initial cooling rate less severe than the exponentially based theoretical cooling rate.

6.2.1.4 – INITIAL COOLING RATE

As a measure of the severity of explicitly described thermal shock, i.e. with thermal gradient and time period, the initial cooling gradient was determined by a simple forward difference method. Figure 44 to Figure 46 show the results of such a measure to be clearly unrefinable. At any level of refined element density for thermal distribution and surface heat transfer coefficient the initial cooling gradient can be seen to refine well. However, with increasing element density the initial cooling gradient is shown to increase in a linear fashion related to the number of elements by unity, and so never refine unless the number of elements is also specified.

A forward difference treatment of theoretical values is shown in Table 22. Again, the same unrefinable rise occurs demonstrating approximation of an initial cooling rate to be indeterminate. Since the theoretical solution is itself the approximation to an infinite series summation, the initial cooling rate can not be used

as a refinement indicator or as a means by which thermal downshock can be explicitly quantified.

6.2.2 – Numerical Finite Difference Solution

6.2.2.1 – COMPARISON TO THEORETICAL SOLUTION

To determine the effectiveness of the finite difference solution in long term convection models a comparison was made to the theoretical solution. Initially transient in behaviour, the thermal distribution will settle in to steady-state, illustrated graphically by changing temperatures but a consistent distribution pattern through time. All transient solutions will settle in to steady-state behaviour, and consequently the finite difference model is run at a given Biot number for time periods spanning four orders of magnitude, 1, 10, 100 & 1000 seconds. Figure 47 shows the effect of a fixed boundary condition at the insulated edge. Temperatures settle into a steady-state form, however such a scenario would require a constant heat flux to maintain the insulated edge temperature when convection effects reach through the entire body of the model.

By applying a further equation allowing heat flow by conduction from internal material to the insulated end, the finite difference model follows further cooling as found in the theoretical solution and illustrated in Figure 48. Consequently it is shown that the finite difference model is capable of accurately modelling the theoretical solution to a one dimensional asymmetrical short and long term shock, given explicit definitions of thermal gradient and time period or simply an initial temperature and surface heat transfer coefficient.

6.2.2.2 – COMPARISON TO SINGLE & DOUBLE EDGE AIR JET SCENARIO

A previous study has provided experimental thermal distributions of single edge and double edge symmetrical shock consisting of cooling a thin plate by small air jets situated close to the plate edge. Air jets were held vertically and activated simultaneously under equal pressure for double edge shock for a given amount of time.

The idealised nature of the finite difference model can be seen in Figure 49 and Figure 50. As the air jets are activated localised cooling takes place at the leading edge and lesser cooling takes place on the plate surface, as turbulent air creates a cooling envelope around the specimen. Initially this results in a gross cooling region ahead of the shocked edge causing a smaller thermal gradient to exist across the plate. As the shock cycle progresses the more intense cooling at the edge takes hold and creates a more severe thermal gradient across the plate. Consequently the finite difference model is found to over idealise the single edge air jet scenario at early stages in the cycle, and is unable to model the small scale global cooling across the plate. However, the majority of the shock cycle thermal gradients at the shocked edge demonstrate a very good correlation.

The introduction of far edge air jet cooling to generate a symmetrical double edge shock and its consequences to the idealised finite difference model is shown in Figure 50. Narrowness of the experimental specimen width causes both air jets to interact and superimpose their cooling effects throughout the surface of the plate. This causes a combined global cooling effect at each edge greater than the effect of single edge cooling modelled at either end. Again a dynamic surface heat transfer coefficient is shown to exist where an initially low level of shock is observed which then accelerates to that shown by the finite difference model in near-edge regions.

6.2.2.3 – TEMPERATURE DEPENDENCE ON THERMAL PROPERTIES

Thermal shock gradients are extreme enough to warrant a temperature dependent analysis of the solution. For the same three levels of shock as conducted above results are shown in Figure 51 to Figure 53. For such a significant change to take place in thermal distribution, large changes must be present in the thermal properties. For AISI 316, thermal properties change by approximately 25% for thermal conductivity and only 5% for density. Since thermal conductivity is the most dependent, coupled with its greater importance in the temperature-dependent finite difference equations, a fairly significant change is observed.

At early stages the general distribution is shown to be consistently similar to that of the fixed thermal property models. At the shocked edge, however, cooling is consistently increased by temperature dependence. As time progresses through the shock cycle, cooling falls in line with the fixed property model as cooling rates

decrease. The general distribution however, begins to cool less rapidly, an effect increasing with time. This is due to the lower thermal conductivity values present at lower temperatures than the maximum temperature properties used in the fixed property model. This should consequently result in slightly higher stresses generated using the temperature dependent model than the fixed property model.

6.2.3 – Finite Element Two-Dimensional Solution

6.2.3.1 – REFINEMENT

Refinement of the finite element solution shown in Figure 54 and Figure 55 is obtained through a short series of models listed by Table 23. The minimum stability condition is such that the relative time increment refines at values of zero. The more positive a value the larger the overall time incrementation, hence the coarser the solution, whereas the more negative the increment the more likely spurious oscillations are to occur in the solution.

Using the maximum/minimum property regime to calculate the largest possible stability condition results in the models capacity to generate a solution inside the unstable region. However, the further into this region the solution goes the more likely the solution is to fail. As time incrementation refines to the minimum increment, surface heat transfer coefficient values reduce towards the theoretical solution.

It appears from Figure 54 that the finite element model is more dependent on time incrementation than that of the finite difference model. With coarse meshes the model determines surface heat transfer coefficients very close to the theoretical solution, when advantage is taken of the ability to run in the unstable region. However highly sensitive dependence is present on time incrementation in these coarse meshes and consequently is inadvisable for general use.

This sensitivity reduces with increasing mesh density until mesh size is almost entirely dominating as in the finite difference model. This is a consequence of the stability condition itself since the smaller the characteristic element length the smaller the minimum time increment. As a result refined meshes inherently cause refined time incrementation.

Using stable time incrementation Figure 55 demonstrates the low mesh density required for refinement of the thermal distribution. Only at the greatest rate of thermal

gradient is the refinement active, showing a final solution with a similar mesh size to that of the finite difference model.

6.2.3.2 – COMPARISON OF TWO-DIMENSIONAL FINITE ELEMENT & FINITE DIFFERENCE SOLUTIONS

Maintaining the highest level of severity of the models refined above, the two dimensional finite difference solution was compared to that of the finite element model. A two dimensional model consists simply of the localisation of a thermal downshock to a particular region as opposed to an entire edge surface. Assuming the cooling is uniformly distributed over the thickness of the model, a cold spot will be generated ahead of the shocked surface and could consequently affect the surface heat transfer coefficient required to attain the explicitly described thermal shock.

Figure 56 shows an almost exact match between both models in distribution and in cooling rate, when exposed to an 8mm localised half-region. However, in order to achieve such a match, the surface heat transfer coefficient must further be manipulated. As shown above, a discrepancy exists in the one-dimensional finite difference model in its ability to converge on the theoretical surface heat transfer coefficient. This manifests itself as a 5-10% error over the theoretical solution and finite element approximation.

No change on the thermal distribution at the symmetry plane is found for localised shock over full edge one-dimensional shock. The effect of localisation on the surface heat transfer coefficient required to achieve the specified shock is shown by Figure 57. For highly localised shock the required surface heat transfer coefficient is almost double that required for full edge shock, an effect more pronounce in the finite element model. However, for localisation above a 10% half-region, the value is maintained at a constant.

Thermal distribution is affected considerably at shocks of high localisation. Figure 58 and Figure 59 show the increased rate of cooling for the finite difference and finite element models respectively. A series of half-regions are shown up to 16mm, equating to a relative height of 0.151. Beyond this point the effect ceases and thermal gradients remain consistent. This is in direct correlation to the point of consistent surface heat transfer coefficients in Figure 57.

When refinement of the model is attempted with highly localised shock, inconsistent surface heat transfer coefficients are determined. Consequently the approximating solutions can not, at this stage be relied upon to adequately model conditions of high localisation. This can only be considered a flaw in the finite approximations where severe solution changes occur, from node to node, at regions highly sensitive to small changes.

6.3 – Stress-Strain Analysis Results

6.3.1 – Linear-Elastic Stress Analysis

The linear-elastic solution as determined analytically by Timoshenko, eqn. 36 [128] is conducted based on a net summation of bulk elastic stresses determined through a cyclic modulus and thermal strains. Thermal strains are here determined from a thermal distribution, calculated via the finite difference solution described in section 5.2.1. This method shall henceforth be referred to as the FD&T method. These can then be used as an independent means of verifying the finite element thermal shock models. It must be noted that due to its net stress summation, eqn. 36 also returns zero thermal stress over linear thermal distributions. Since thermal shock is a non-linear distribution this does not present a problem. However, eqn. 36 is a one-dimensional solution, and as such does not account for multi-dimensional effects, and is likely to inadequately describe the total strains taking place. It cannot therefore be used to analyse the effects of localisation of the shocked region. It can be used well, however, to verify the linear-elastic and elastic-plastic models of the finite element models.

6.3.1.1 – FULL EDGE SHOCK

Figure 60 through Figure 62 show typical thermal shock stress profiles through the shock cycle. Stresses are calculated from eqn.36 with fixed and temperature dependent elastic thermomechanical properties. A maximum temperature elastic modulus and thermal expansion coefficient is used for the fixed property calculation. Very little difference is determined between the two and since thermal gradients exist, approximately in the first 15mm, stresses are closely matched ahead of this region. However, prior to this region stress gradients can be noticeably larger, as the severity of thermal stress at the surface is increased and the penetrative distance of the first tensile region is reduced.

A comparison of end-of-cycle stresses is made for the three levels of shock between FD&T and finite element temperature dependence. Illustrated in Figure 63 and Figure 64 an excellent relation to both models is determined for fixed property analysis. However finite element demonstrates higher stresses and gradients for the

same thermal distribution and properties as determined analytically by the FD&T solution.

It must be noted at this point that the overall change in elastic modulus and thermal expansion is approximately 10% and 15% respectively. Therefore, material with a greater susceptibility to thermal changes is likely to encounter a greater reduction in tensile region, and much higher levels of stress at the surface over a fixed property analysis.

Lower maximum stress levels and a much broader spanning of stresses through time indicate the reducing severity of thermal shock loading. This is illustrated in Figure 65 showing elastic stress at the shocked surface for both finite difference and finite element analyses determined from a fixed cyclic modulus and thermal expansion specified at the maximum temperature. It is apparent the large changes in temperature at the initiation of the downshock cycle is reflected by sudden changes in stress in the near surface region. These high gradients are seen to ease off as the cycle proceeds, and for high downshock loads begin to fall off at the end of the cycle. This results in the further consideration of the maximum stress state to be at some intermediate point in the cycle as opposed to the end, consequently affecting the classical definition of the stress intensity range.

In these cases of high levels of shock, a point exists in the model at which stresses cease to reduce in the later stages of the cycle. This can be shown to be a direct result of the rate of heat extraction by forced convection exceeding the rate of heat flow by conduction and will be discussed in detail in a later section.

Figure 66 shows the effect of introducing temperature dependant cyclic modulus and thermal expansion coefficient. The solids lines indicate the fixed property analytical stresses of Figure 65. Using the same properties for the finite element solution and FD&T solution, considerable differences in the time distribution are found at high levels of shock. This discrepancy is not considered a result of the two-dimensional thermomechanical strains present in the finite element model, since it would be displayed in Figure 65. The greatest difference lies at the low temperature end-of-cycle region for the 625-225°C in 3s shock. It is also in this temperature region of 0-300°C that changes in cyclic modulus are coarsely defined due to the lack of information for AISI 316 at these temperatures. Both FD&T solution and finite element solution conduct a linear interpolation from quoted temperatures. Since

properties are maintained between models and methods of property interpolation are consistent, it must be concluded two-dimensional effects are enhanced under changing material properties.

For the 650-350°C in 5s shock, the small oscillation in stress determined by the FD&T method early in the cycle takes place in the range of 450-550°C. From higher temperatures the cyclic modulus slowly rises, with a sudden drop taking place around 500°C to slowly increase again with decreasing temperature. Consequently, a small drop in stress is observed, continuing into higher stresses than the fixed property profile. This is not observed in the two remaining models, since the 450-550°C range exists very early in the cycle; approximately before 0.2s. The fact that this does not take place in the finite element models demonstrates further difference on a two-dimensional stress field, accounting for mechanical as well as thermal strains.

6.3.1.2 – BIAXIAL EFFECT OF LOCALISATION

Although little effect of localising the thermal shock region is observed in thermal loading and distribution, the effect on stress distribution is significant. Figure 67 and Figure 68 illustrate the effect on the end-of-cycle stresses for the most severe shock of the three discussed.

Primarily when the shocked region is reduced, biaxial stress fields are generated ahead of the shocked surface. Maximum tensile and compressive normal stresses are seen to reduce as the amount of contraction desired at the surface is reduced. However lateral stresses do not follow the same behaviour. Because localisation allows relative contraction to take place parallel to heat flow, compressive stresses are forced into the plate as the shocked region contracts inwards, generating further compressive stresses through the plate. With no lateral mechanical loads, this stress field starts and ends at zero, with an internal compressive region initially zero for full-edge shock. As localisation increases the compressive lateral stress increases in magnitude and width of distribution. As localisation becomes very small the compressive stress decreases only slightly, as is illustrated in Figure 69. The maximum compressive lateral stress clearly occurs at 8mm (16mm overall due to symmetry) or 7.5% of full edge shock. This occurs at position very close to the maximum normal compressive stress, illustrated in Figure 70. Here, the point of maximum compressive lateral stress moves away from the crack tip, only to return to very much its original position. Hence, there

exists a distinct level of localisation where the two normal and lateral maximum compressive stress states can combine in close proximity to the crack tip. The level of localisation coincides with the experimental work used here from a previous study. Thermal downshock was achieved via an air jet 16mm wide creating a very similar region of localised thermal shock. Crack tip blunting could be due to a compressive lateral stress and therefore dampen the crack closure caused by the compressive normal load leading to increased, though slow crack propagation than otherwise believed under a full edge thermal shock analysis.

It is interesting at this point to hypothesise that the inverse could occur for thermal upshock, where tensile normal and lateral component stresses may combine. Under such conditions crack propagation may be slowed due to the lateral tensile stress inducing crack tip closure.

Figure 71 and Figure 72 demonstrate the evolution of the maximum compressive normal and lateral stresses through time respectively. Normal stresses show a steady reduction as the shocked region reduces, with a rapid rise to a maximum for small regions. As the shocked region increases the maximum compressive stress state lays more toward the end of the shocked cycle, requiring significantly longer cycles in order to observe any level of stress reduction. This is not mirrored however in the lateral stresses. Again, initial stresses rise quickly but are significantly reduced by the end of the shock cycle when localisation is small. With increasing size of the shocked region, lateral stresses reduce in magnitude and rate, leaving a maximum state at around an 8mm half region, i.e. a 16mm shocked region in practice.

To describe the resulting biaxiality three biaxial parameters are plotted in Figure 73. Parameters are described below.

$$B_1 = \frac{\sigma_L \sigma_N}{\sigma_{yield}^2} \qquad B_2 = \frac{\sigma_L}{\sigma_N} \qquad B_3 = \frac{2\sigma_L \sigma_N}{\sigma_L^2 + \sigma_N^2}$$

eqn. 84

Functions B_1 and B_3 have been derived here in an attempt to avoid the singular event of normal stresses passing through zero. In this case the standard biaxial parameter, B_2, is caused to rapidly accelerate to infinity. Using B_1, which introduces a control such as a yield stress, and maintaining both normal and lateral stresses as nominators can alleviate this. This creates the linear comparison of biaxiality, whilst also introducing a weighting effect of the overall stress level in relation to yield. B_3 is a second order function which might be introduced to maintain a zero value for either

zero normal or lateral stress, whilst maintaining a unity value for normal and lateral equality. The function still does not behave linearly and consequently overestimates the standard biaxial function of B_2.

It is clear from Figure 73 that a high level of biaxiality is present in the compressive region of the model. However the behaviour of a crack in this region is not easily predictable and certainly unlikely to propagate along the symmetry plane in this region. At the shocked edge only a slight level of biaxiality is found because no mechanical loads are applied in this region, though both B_2 and B_3 parameters show a very rapid increase toward the start of the compressive normal region. As expected, the second order function overestimates the standard B_2 parameter though it does mirror the parameter well whilst never exceeding unity. The weighted parameter shows biaxiality in the same places as the standard parameter, B_2, whilst also indicating the low levels of lateral stress in relation to a yield stress.

The three lines of constant value represent the predefined levels of biaxiality present in the unibiaxial tests previously conducted. With equibiaxiality represented by a value of 1 for functions B_2 and B_3 the lines show bounding limits of biaxiality for the isothermal load cases on thermal shock. For function B_1, the equibiaxial value is dependent on yield. However, it will be higher than the unibiaxial line. Consequently, the B_1 function does not bound the thermal shock levels of biaxiality at all, although, functions B_2 and B_3 are more favourable. Both B_2 and B_3 unibiaxial functions pass through the approximate average of the biaxial profile of thermal shock. This indicates that the thermal shock biaxiality, localised to that of the previous experimental study, is partly bounded by the unibiaxial study. The crack growth rates of these two scenarios, isothermal-biaxial and localised thermal shock, are therefore shown to coexist over the similar biaxial ranges.

To extrapolate this to its furthest extent, the fully bounding condition of unibiaxiality can be read from Figure 73 as approximately 0.35 for the standard function B2 and approximately 0.65 the second order function B_3. It is then a question of determining the loading magnitudes to create the best comparison between isothermal-biaxial and localised thermal shock crack growth rates.

6.3.2 – Strain Analysis

Using the finite difference thermal solution, thermal strains are calculated from αT for an elastic stress analysis. Thermal strains by finite element are calculated by eqn. 85.

$$\varepsilon_{th}(\theta) = \alpha(\theta) \cdot [T - T_{REF}] - \alpha(\theta_I) \cdot [T_I - T_{REF}]$$

eqn. 85

A reference temperature, T_{REF}, is provided only for temperature dependant thermal expansion, α. This allows the second term to remove any thermal strains incurred by initial conditions, T_I, different to that of T_{REF} allowing initial conditions to represent zero thermal strain; consequently thermal strains for a fixed property finite element model also uses αT.

Further to this, finite element methods allow strains to be calculated which account for the presence of a two-dimensional thermal field and bending caused by asymmetrical thermal distributions. These total strains illustrate the effects of localisation on stress distributions but do not show a distinct enough difference on fixed property to temperature dependent analyses to explain the large differences observed in 625-225°C in 3s shock.

6.3.2.1 – TOTAL STRAIN

Figure 74 illustrates the simple thermal strain calculation by αT. Little difference is observed between temperature dependent and fixed property analyses, as would be expected for the slight changes in thermal expansion taking place. Figure 75, however, demonstrates something more interesting. Total lateral strains are approximately equal to thermal strains near the shocked edge, with a fall off in strain at the far edge. This fall off is due to the significant bending effect taking place throughout a full edge asymmetrical shock. As the far edge goes into normal-tension, Poisson's ratio withdraws material laterally causing a reduction in tensile strain. This results in a slight drop in strain at the far edge, proportional to the size of the bending moment. This is supported by Figure 76 where normal strains are linearly distributed through the model. At the shocked edge these strains fall off through the cycle as temperatures recede, at the far edge however, strains increase as the bending moment increases

through time. A comparison of end-of-cycle strains is illustrated by Figure 77. Unlike stresses, orthogonal strains are seen to exist within the same magnitude even at full-edge shock levels.

6.3.2.2 – EFFECT OF PLASTICITY

Using a deformation plasticity model to follow the full stress/strain behaviour determined by Neuber's relation, the effect of plasticity on mechanical strains can be determined. Figure 78 illustrates the presence of larger levels of normal strain in the plasticity model in near shocked regions for the full-edge shock model of Figure 77. Such an increase is likely to cause an increase in magnitude and a broader spanning of the near-edge tensile region of a stress distribution. Introduction of plasticity generates lower stresses for a given strain, this causes lower bending stresses, and consequently a more shallow normal strain distribution. This is shown by Figure 79 where decreasing the level of shock severity shows a reduction in the gradients of normal strain. The higher mean value of the 650-350°C in 5s shock is not a result of higher shock severity, but of higher bulk temperature. This would lead to the assumption that normal strain for full-edge shock under symmetrical conditions would be constant. Localisation of the shocked region supports this, whilst symmetrical double edge shock is discussed in a later section.

6.3.2.3 – EFFECT OF LOCALISATION

When the applied forced convection is localised to a partial region of the plates edge similar changes take place in strain distribution to that of the stress distribution. Illustrated by Figure 80, lateral strains show only slight changes over large increases in localisation. The most prominent effect being that of the models with a larger surface area exposed to forced convection, i.e. 27mm and 53mm. Here a larger mid-section strain and lower far edge strain is shown, undoubtedly a result of a higher level of bending. This can be explained where small shocked regions cause the bending effect to be significantly reduced as only a small surface area is attempting to contract. This in turn will generate a much smaller and localised bending moment at the far edge.

Again this is supported by Figure 81. At small exposed surface areas normal strain distribution possesses much the same form as that of the lateral strain. This is

due to the form of the normal thermal distribution being very similar to the lateral thermal distribution. With increasing levels of shocked surface area the effect of bending moment on strain becomes more prominent; less thermal gradient exists normal to the symmetry plane and strains fall in the near shocked region, only to be increased by the rising bending moment at the far edge.

6.3.3 – Elastic-Plastic Stress Analysis

Elastic analyses produce very high stresses at the shocked surface. Since such stresses are greatly beyond any yield stress of the material it is appropriate to quantify plasticity effects on the stress distribution. It is quickly apparent that for an absolute strain, αT, thermal strains are already above a strain corresponding to yield stress at temperatures over 100°C approximately, therefore the elastic strain region will be bypassed entirely. A more appropriate thermal gradient analysis using $\alpha[T-T_I]$ is performed with interesting results.

6.3.3.1 – FULL-SINGLE-EDGE SHOCK

Using the FD&T method to perform an elastic stress analysis, it is found that the solutions for thermal strain calculated by both αT and $\alpha[T-T_I]$ are identical. This is a result of the net effect consideration of eqn. 36, from three separate stress terms, where the relative difference between these terms is used to determine the stress distribution. Since αT and $\alpha[T-T_I]$ describe the thermal distribution from a constant reference temperature the results are consequently equal. This equality is illustrated by Figure 82 where a cyclic elastic modulus is used. The result of conducting elastic-plastic analyses with αT is also illustrated for a full Neuber's (EP) stress/strain material response and a partial linear-elastic, elastic-plastic (LE-EP) response. Very low unlikely stresses are a result. Both methods equate since αT determines no strains in the linear region, and so differences due to linear analyses do not appear.

Using the thermal gradient for determining thermal strains, the same two plasticity models, EP and LE-EP, are conducted using the FD&T method and compared to FE results. Figure 83 illustrates the first; a full elastic-plastic relationship is modelled by Neuber's equation embedded into the FD&T solution. In comparison to

the cyclic modulus elastic analysis the differences are extreme, the tensile and compressive regions are reduced in magnitude by 80% and 50% respectively and the width of the tensile region is significantly increased by over 30%. The result being a very good match between elastic FD&T and FE stress distribution. However when the same plasticity model is introduced to a FE model the stress distributions are markedly different. This is a result of the inherent biaxiality present in plasticity.

Further to this, a partial LE-EP model was conducted using a pseudo mechanical modulus below an equivalent yield strain and the standard Neuber's relationship above the equivalent yield strain. By conducting an elastic analysis at this pseudo mechanical modulus, a comparison can be made between the partial LE-EP and the linear-elastic model. Figure 84 demonstrates the results. Again a very high reduction in stress is gained for the LE-EP model similar to that of the full Neuber's EP distribution. Interestingly, however, the mechanical LE stress distribution illustrates a very close correlation to the LE-EP distribution, diverging only around the specified yield stress. This is a result of the difference in strain energy over the given thermal strain range when using $\alpha[T-T_I]$ as mentioned previously in materials analysis, Chapter 4.

The relative change in spanned area between the elastic and elastic-plastic material responses is very similar to that of the relative change in absolute area spanned by the resulting thermal shock stress distribution. Table 24 presents an analysis of area ratio for the four different material models used for three different levels of shock.

Strain energy represents the area beneath a stress-strain (σ-ε) material response and is calculated from the material model which determines a bulk stress for a given maximum thermal strain. The ratio of elastic strain energy by cyclic modulus to Neuber's EP strain energy is given as the *cyclic ratio*. Similarly the ratio of mechanical modulus to LE-EP strain energy is given as the *mechanical ratio*. For comparison this is also done with the absolute spanned areas of the stress distribution (σ-x) through the symmetry plane. The pseudo-mechanical model for 550-350°C in 5s generates thermal strains just below that corresponding to a 0.2% proof stress. Consequently no plasticity is introduced into the LE-EP model and a spanned area ratio of unity is determined. A unity ratio is not determined for the strain energy quantity due to the 0.2% proof stress introducing a small error in the linear-elastic to elastic-plastic transition.

The result is a very good correlation between spanned area of resulting stresses and total strain energy for the three levels of shock shown. Illustrating materials with a reasonably high level of ductility will require a plastic stress analysis to correctly determine near shocked edge stresses.

Finite element analyses using a deformation plasticity model to follow the same stress/strain response as that of Neuber's, produces stresses significantly different from that of a one-dimensional thermal strain analysis by the FD&T method. The presence of mechanical strains increase the overall stresses induced from thermal strains. This effect is increased in the presence of plasticity, where bending stresses reduce but normal strains increase. Consequently, accounting for thermo-mechanical strains, larger stresses are created in an EP analysis than a similar EP analysis by the FD&T method accounting only for thermal strains.

6.3.3.2 – DOUBLE-EDGE SYMMETRICAL SHOCK

Three major changes occur when a symmetrical thermal shock is generated. All are related to the changes in bending moment taking place. Figure 85 shows the bending moment displacements of the neutral axis determined by the finite element model with varying degrees of localisation. Also included are the theoretical bending displacements for a simple beam with a bending moment at either end. As proposed in previous sections, bending is reduced both by the introduction of plasticity and localisation of the shocked region. As expected when symmetrical conditions arise, no bending occurs for full-edge or localised shock.

Increasing localisation creates further differences between the bending observed and the classical solution. This is due to the non-uniformity of bending across the model when shocks are localised. Full-edge shock performs as an end moment, however, with localisation of shock, loading becomes more similar to a distributed load in that region.

Analysis of lateral strain renders very little difference than that expected when compared to single-edge as illustrated by Figure 86. However, as proposed at the end of section 6.3.2.2, the normal strain shown by Figure 87, behaves similarly to that of single-edge shock until large shocked areas are used. As the shocked surface area increases the normal strains do in fact level out to a constant, due to the absence of bending.

Finally, stress analyses show an elevated level of stress for double-edge shock over single edge illustrated by Figure 88. In elastic analyses this increase is even greater, and is considered to be a consequence of the levelling out of normal strains, where single-edge strains are significantly lower in the near edge region. Lateral stresses are also increased more significantly than normal stresses shown by Figure 89. Since lateral compressive stresses are caused by localised contraction of near edge material, it is inevitable that double-edge shock will generate a larger compressive region as opposing regions contract against each other.

This could create an exaggeration of the Tomkins shear decohesion model of crack propagation. With increased tensile stress at near shocked regions, the length of the tensile region is increased, allowing cracks to propagate further. With additional compressive lateral stress the effect on crack tip blunting would be marked. As a result, the possibility of crack tip spalling to regions of lower normal tensile stress could occur, pushing the crack away from the compressive region. The further possibility arises that opposite edge cracks, generating their own small tensile stress fields ahead of the crack tip, could spread around the compressive zone and merge to cause catastrophic failure. Taking place under large widths will likely suppress any such activity. However, with small widths, such as those of the thickness of pressure vessels, could be vulnerable to a spalling event and further in-depth study on the likelihood of crack tip spalling under symmetrical shock must be recommended.

6.4 – Fracture Analysis Results

6.4.1 – Isothermal Cruciform Analysis

6.4.1.1 – LINEAR-ELASTIC STRESS INTENSITY FACTOR

A mode I stress intensity range describing the cruciform specimen crack tip loading behaviour is taken from Brown and Miller [29] and given below in eqn. 86,

$$\Delta K_I = \Delta\sigma \sqrt{\frac{\pi a}{\cos\left(\frac{\pi a}{W}\right)}}$$

eqn. 86

where $\Delta\sigma$ is the applied stress range normal to the crack and W is the gauge length of the central square of the cruciform. As is the nature of the stress intensity, it is immediately apparent that no effect of biaxiality is accounted for; specifically the non-singular term for T-Stress governing biaxiality plays no part of the stress intensity. This is illustrated by the dashed lines of Figure 90, where a set of crack lengths for 4 separate loading conditions, quoted from crack growth data made available from a previous study [124], are used to determined the respective ΔK_I. The 4 loading conditions were applied in-phase in equibiaxial and unibiaxial manners at stress ranges of 118.8MPa and 200MPa with an R-ratio of 0.1. Unibiaxial loading consisted of oscillating the normal load whilst holding the lateral load at a constant stress equal to the mean stress of the normal load.

A set of finite element models were also conducted with ΔK_I values determined from the closed contour integral J by determining K_I values for the peaks of a sinusoidal load form and determined through eqn. 87.

$$\Delta K_I = \sqrt{J_{MAX} E} - \sqrt{J_{MIN} E}$$

eqn. 87

These are indicated by the single point values of Figure 90 correlating almost exactly with theoretical values. As is expected there is no change in result when T-Stress is changed from equibiaxial to unibiaxial loading.

The FE models of Figure 90 were conducted at a uniform thickness equal to that of the central square plate. However, when the thickness is modelled in a plane solid section, the effect on the stress intensity is marked.

Crack growth data was available for cruciform specimens with a central plate thickness of 4mm and a loading fin thickness of 15mm. Figure 91 illustrates the effect. Three sets of data are determined; theoretical of eqn. 86, finite element models by contour integral and finite element by crack tip opening displacement.

The two methods conducted by the finite element models are in excellent agreement, however, they are far removed from the theoretical. This is due to the theoretical model not accounting for the change in stress encountered at the loading fins and plate intersection. A stress is applied at the loading fins and is increased by the reduction in area at the intersection of the loading fins and plate. Since it is more appropriate to describe the applied stress on a crack tip as that undisturbed by any discontinuity, the stress at the plate edges must be used, and not the stress at the loading fins.

Therefore, the applied stress range, $\Delta\sigma$, must be calculated from the applied load and plate section area or by applying a geometry factor t_F/t_B to eqn. 86, where t_F and t_B are the loading fin thickness and the central square plate thickness respectively.

$$\Delta K_I = \Delta\sigma \frac{t_F}{t_P} \sqrt{\frac{\pi a}{\cos\left(\frac{\pi a}{W}\right)}}$$

eqn. 88

Illustrated in Figure 92, the modified theory falls in line with the finite element models almost exactly.

6.4.1-2 – EFFECTIVE STRESS INTENSITY

To determine the effect of plasticity on the FE fracture models, elastic-plastic material properties were introduced and a ΔK_I determined from eqn. 87. Although such stress intensity is theoretically invalid, it does illustrate predictability almost as rigorous as the linear-elastic results. Figure 93 shows the effective ΔK_I to be slightly less than the linear-elastic. This is due to a decrease in strain energy around the crack tip resulting in lower contour integrals. Stress intensity factors are presented for all four loading conditions for both linear-elastic (LE) and elastic-plastic (EP) properties

with uniform thickness through the model set at the plate thickness (UT). Results for elastic-plastic models at longer crack lengths and a stress range of 200MPa were unable to be obtained. This was due to convergence difficulties around the crack tip, caused by the onset of collapse due to the high ductility of the material. Neither could non-uniform thickness models be conducted with elastic-plastic properties for the same reasons, though the increase in stress at the plate due to the change in thickness was the primary cause.

Curves can be fitted to the effective stress intensity using eqn. 86 and altering $\Delta\sigma$ to $\Delta\sigma'$ indicated on the figure and approximated by eqn. 89 where n is the strain-hardening exponent of a Ramberg-Osgood material i.e. $0 < n < 1$.

$$\Delta\sigma' \approx n\Delta\sigma$$

eqn. 89

To create a complete model with elastic-plastic properties and non-uniform thickness a three-dimensional model would be required with radii included at the loading fin to plate intersection. Such a model would require a further programme of study to evaluate the effects of the intersection radii. These would likely control the models capability to converge creating similar problems already encountered in the two dimensional model. As a result such a model was considered to be outside the time scale of this work.

6.4.1.3 – SUMMARY OF CRACK GROWTH RATES

Crack length to number of cycles data for isothermal cruciform tests from a previouds study is analysed by using ASTM E647 crack growth analysis model. Crack growth rates have been calculated and plotted with stress intensity factors from the linear-elastic material and elastic-plastic material models. Figure 94 shows the correlation of crack growth rates to the stress intensity range determined by a linear-elastic uniform thickness analysis for the four loading conditions described above. Stress ranges of 118MPa show a good distribution where equibiaxial loading has significantly reduced crack growth rates, however, at higher stress ranges of 200MPa the data is less coherent. Unibiaxial data at a stress range of 200MPa shows a distinctive line higher and steeper than that of lower loads. For equibiaxial loading at 200MPa, crack growth rates experience more scatter but show a definite increase in crack growth, approximately equal to the decrease experienced at the lower 118MPa

stress range. At higher stress intensity ranges, a sudden drop occurs making crack growth rates approximately equal to those of the unibiaxial data.

Figure 95 shows the correlation of crack growth rates to the stress intensity range determined by elastic-plastic uniform thickness models. The same correlation is found as in Figure 94 with a slight shift to the left, resulting from elastic-plastic material properties. Figure 96 shows a comparison of the two. With the inclusion of elastic-plastic material properties, it can be seen that the crack growth rates are significantly underestimated when using linear-elastic material properties. Since the crack growths rates are calculated directly from a-N data, a typical reduction in stress intensity of around 30% from elastic-plastic material can result in crack growth rates 70% higher than that described by linear-elastic material models.

6.4.2 – Theoretical Plastic Zones for Linear-Elastic Materials

6.4.2.1 – COMPONENT STRESSES

The theoretical plastic zone parameter derived in a previous section and reiterated below in eqn. 90 for convenience is here evaluated as a viable parameter.

$$\frac{\Delta r_{PvML}}{\Delta r_{PvMH}} \cdot \Delta r_{Py}$$

eqn. 90

In order for the parameter to be viable, two conditions must be satisfied;

- Must describe effect of normal load magnitude independent of σ_T.
- Shape ratio must be constant and unique over a for full range of T-stress.

Stress intensity factors from the above finite element isothermal models have been used in conjunction with the four different loading scenarios described above. Theoretical plastic zones from component stress functions (normal, N, and lateral, L, to the crack plane) are illustrated at several crack lengths in Figure 97. Specifically the normal plastic zone sizes are equal for equibiaxial and unibiaxial loading. This satisfies the first of the two viability conditions for the plastic zone function. Equibiaxial lateral plastic zones are not illustrated in the figure since they are by definition equal to the equibiaxial normal plastic zones. Figure 98 shows the ratio of change in these values as would be used under the von Mises shape ratio. The ratio of the two component stresses at equibiaxial loading shows to be, predictably, unity.

The main feature to be taken from this figure is that as predicted all crack lengths conducted show a constant and unique ratio for different biaxial loading. It is therefore likely that a von Mises plastic zone shape ratio would show the same characteristic.

6.4.2.2 – VON MISES STRESS

Figure 99 shows the length (L) and height (H) of the theoretical linear-elastic von Mises plastic zone accounting for σ_T. A primarily linear relationship is found in relation to increasing crack length. As loads, however, are increased the relationship develops a slight parabola. There is also present a shift in ordering of the length and height of von Mises plastic zone. Whilst the size of the stress range always holds priority, the length of the plastic zone is favoured by equibiaxial loading and the height is favoured by unibiaxial loading. This means that σ_T is directly affecting the shape of the von Mises plastic zone with loading magnitude maintained as a separate effect on size.

The shape ratio for the change in a von Mises plastic zone is plotted in Figure 100. As predicted, the ratio is constant and unique for separate biaxial conditions. A value of unity is not present for equibiaxial loading since the von Mises plastic zone is not circular, hence the length and height are not equal. Coupled with Figure 99 a complete description of the von Mises plastic zone, satisfying the two required conditions, is presented in Figure 101. A very similar plot is obtained to that of Figure 97 and Figure 99. However, in this case the four loading conditions are identified entirely by a single parameter. Loading magnitude is favoured such that high loads create proportionally higher parameter values with equibiaxial loading than unibiaxial. This is as required since equibiaxial loading creates greatest effect on the plastic zone.

By way of demonstrating the shape ratio's capability to uniquely describe the applied biaxiality, Figure 102 is presented. It shows the relationship of shape ratio to the ratio of σ_T to yield stress for all values of σ_T within yield stress. Positive values indicate a lateral stress greater than that of the normal load and visa-versa for negative values. Several limiting points are illustrated here. Firstly, a kink occurs at around $0.75\sigma_{yield}$. This is due to oscillation in the change of the plastic zone shape. As the shape changes with increasing σ_T, '*tail fins*' develop behind the crack tip, these are

larger in height than the zone ahead of the crack tip. Since the parameter is describing only mode I non-inclined cracks, only the maximum height ahead of the crack tip is taken. Under mixed mode loading the plastic zone becomes much more complex and would require an account of these tail fins and their orientation to the crack plane. Secondly, the relationship shows that for σ_T equal to ±yield an infinite solution is obtained. This would require, however, that the solution be passed through an unverifiable region. The curve begins and ends at values of σ_T/σ_{yield} equal to -0.617 and +0.893 respectively. Outside this region the quadratic check on the solution fails and an appropriate solution can not be found.

6.4.3 – Finite Element Plastic Zones for Elastic-Plastic Materials

6.4.3.1 – COMPONENT STRESSES

Using elastic-plastic properties, finite element models for the above scenarios have been conducted. Figure 103 and Figure 104 mirror the theoretical linear-elastic relationships presented in Figure 97 and Figure 99. However, Figure 103, shows only the peak load plastic zone as minimum load plastic zones showed small compressive regions in the first element ahead of the crack tip and no coherent shape outside that of the element itself. For this reason the minimum component plastic zones were not considered. Though a linear relationship exists primarily for lateral plastic zones and low loads the relationship becomes distinctly non-linear for normal plastic zones and high loads. The size of the plastic zone, however, is also increased significantly to the same order as the crack length. This may result in an inadequate description of the near tip stress field.

The shape of the plastic zones has also changed. Under equibiaxial loading the lateral and normal plastic zones are no longer equal in length, the lateral is smaller as if in the presence of negative σ_T. Lateral plastic zones have also lost the fish tail form behind the crack tip. These effects are all due to the phenomenon of the *plastic mixity* parameter, produced in the non-linear elastic theory of the stress intensity factor. First calculated by Shih in 1974 plastic zones for various strain-hardening exponents, n, were shown. As n is increased the shape of the plastic zone protrudes forward, increasing in length and reducing in height. A consequence of this is that the shape

ratio of the proposed plastic zone parameter may not remain constant or form a unique description of σ_T.

It must also be noted at this point that the normal plastic zone shape may no longer be independent of σ_T. This is used in the first viability condition of section 6.4.2.1. Illustrated in Figure 105 and Figure 106 theoretical component plastic zones are compared with the finite element plastic zones using elastic-plastic material properties for equibiaxial and unibiaxial loading respectively. It can be seen that the normal plastic zone for equibiaxial loading is slightly longer and taller than that of the unibiaxial version. Since these changes are relatively small compared to the changes in lateral plastic zone size the effect is confined to plasticity. As lateral strains will partially restrain normal strains through Poisson's ratio, it is likely disproportionate increases in plastic strain are generated for increases in lateral load. Whereas for linear-elasticity, changes in elastic strain are consistently proportionate to changes in load, therefore plastic zones will not be adversely effected.

The ratio of normal to lateral component plastic zone sizes at maximum load is given in Figure 107. When compared to Figure 98 the results are not promising. Though for three out of the four loading scenarios an approximate constant is still evident, the equibiaxial 200MPa load possesses an extreme downward trend. A further point is the significant inequality of the two equibiaxial ratios. This infers that the von Mises plastic zone shape ratio may no longer uniquely describe the biaxial state.

6.4.3.2 – VON MISES STRESS

Figure 108 shows the shape ratio for the full change in von Mises plastic zone. In contrast to Figure 107 the results are quite promising. The ratio is maintained at a good approximate constant and shows the same order as demonstrated in the theoretical linear-elastic material in Figure 100. Though the equibiaxial loading states do not show equality in shape ratio as in linear-elastic theory, it must be resigned that elastic-plastic material will create this effect due to the plastic mixity parameter mentioned above. As a consequence it is also likely that no uniqueness in description will be lost to this effect, since the mixity parameter is due to the material only and not to differences in applied loads.

Using the normal component plastic zone at peak load only, Figure 109 shows the von Mises plastic zone parameter accounting for σ_T. The relationship is very much

like that of the linear-elastic relationship in Figure 101 but with order of magnitude increases, and increased non-linearity at high loads. The relationship also holds very well, describing the four separate loading conditions as uniquely as that of the linear-elastic description.

Since the change in plastic zone size is an integral part of the loading description, the linear-elastic value of the normal component plastic zone was included to determine its approximate consequence. Figure 110 shows the complete von Mises plastic zone parameter accounting for σ_T with the linear-elastic minimum normal plastic zone. Little difference actually takes place, only a slight reduction in magnitude occurs, with the relationship becoming more linear than non-linear. Since a change nonetheless occurs, however, it is prudent to perform the same check on all profiles, especially those with greater non-linearity such as those for thermal shock.

6.4.4 – Thermal Shock Analysis

6.4.4.1 – EFFECTIVE STRESS INTENSITY FACTOR

Fracture analyses of various thermal shock scenarios have been conducted for both single-edge and double-edge shocks, with and without a mean applied load. Using elastic-plastic material properties and eqn. 87, effective stress intensity factors have been calculated. As previously discussed the maximum load occurs very much at the end of these applied shocks, and it is this end-of-shock stress intensity which is plotted in Figure 111. The general trend is that of the expected rise and fall in the stress intensity due to the crack tip encroaching on the compressive field. Application of a mean load of similar size to that of the maximum compressive stress can, however, result in continued rise of the stress intensity [124]. The shocks presented here are particularly severe, resulting in a considerably high compressive field. The effect of shock severity on the magnitude of this compressive stress is very similar to that of the tensile stress found at the surface, consequently for less severe shocks the mean load required to override the fall in stress intensity will be proportionately smaller.

This can be seen in the double-edge shock data with a mean load of 75MPa. The magnitude of the compressive region is in the region of 75-100MPa making the mean

load just under that required. Consequently the stress intensity levels off, peaking at around twice the crack length of a zero mean load and slowly falling off.

Stress intensity ranges, however, are not effected in this way as illustrated in Figure 112 where all shocks fall off. This is a result of residual strains collecting ahead of the crack tip. As crack lengths increase higher plastic strains are generated, leading to higher residual strains ahead of the crack tip. Consequently, the minimum stress intensity changes at a lesser rate than that of the maximum, producing larger decreases in stress intensity range at higher crack lengths. This is a vital factor in the correlation of crack growth rates for thermal shock. The maximum stress intensity oscillates over a range of around $10MPa.m^{-3/2}$ for individual shocks, and around $30MPa.m^{-3/2}$ for the range of shock levels shown, whilst the stress intensity range oscillates at only half this. When compared with stress intensity ranges for isothermal cruciform models this is a much higher range at early crack lengths, but isolated to a distinct mid range band. Consequently, correlation to crack growth data generates very steep gradients over a considerable range of crack growth rates. The plastic zone parameter proves no exception to this.

6.4.4.2 – THERMAL SHOCK VON MISES PLASTIC ZONE WITH T-STRESS

The plastic zone parameter for the above thermal shocks is illustrated in Figure 113 and Figure 114. Figure 113 shows the parameter using only the normal plastic zone size at peak load. Very similar profiles to those of the end-of-shock stress intensity factor are obtained indicating the plastic zone parameter is appropriately defined. Again mean load models show a distinctive rise in magnitude and a reduction in the tendency for the parameter to fall at longer crack lengths. Applying the minimum size of the linear-elastic normal plastic zone, calculated from the minimum effective stress intensity factor, the plastic zone parameter should experience the same effect as the stress intensity range, i.e. the continued rise due to a mean load should disappear. This is indeed the case as illustrated in Figure 114.

The range over which the plastic zone parameter exists for these thermal shocks is proportionally greater than that of the stress intensity factor and range. However, the range is much reduced when compared to the range for isothermal cruciform values illustrated in Figure 115. Consequently, correlation to crack growth rates will likely create equally steep gradients to those of a stress intensity range. When compared to

isothermal cruciform values it is immediately obvious that the plastic zone parameter is far more sensitive to changes in loading scenario. Here the plastic zone parameter shows under thermal shock conditions to be highly suppressed in relation to the high values of the cruciform results. This does, however, present some intriguing results; at short crack lengths where loading on the crack tip is highest the thermal shock plastic zone parameter follows the high loading cruciform scenarios. As crack lengths increase and loading around the crack tip falls, the thermal shock plastic zone parameter moves away toward the low loaded cruciform results. This would indicate the plastic zone parameter is a better correlating parameter for thermal shock and isothermal biaxial loading than the stress intensity range.

Figure 114 shows some overlapping of results around the peak values and some uncharacteristically high scatter for numerical solutions. This is due to the method of plastic zone data extraction from the models. Though selection of these magnitudes is quite accurate, some error is inevitable due to the plastic zone post-processing programme calculating co-ordinates of integration points without extrapolating data to the nodes. Consequently, any part of the plastic zone passing close to the edge of an element, i.e. outside the area spanned by the integration points, are missed. This results in further approximation which, whilst only minor, accumulates to relatively large scatter over the six values required for the full parameter's description. The fitted lines, however, do demonstrate a good representation to the response of the parameter.

6.4.5 – Correlation of Isothermal Biaxial & Thermal Shock Crack Growth Rates

6.4.5.1 – PLASTIC ZONE PARAMETER

Figure 116 to Figure 118 show the final correlation of thermal shock crack growth rates to isothermal biaxial crack growth rates using the von Mises plastic zone parameter. Figure 116 shows the isothermal crack growth rates, and demonstrates the ability of the plastic zone to be related to isothermal crack growth rates. The parameter is, however, less tolerant of data scatter. Whilst the 118MPa stress range data still holds a good relationship, the quality of 200MPa data has been reduced further. Whilst the general trend is certainly still present increased scatter has made identification of

the distribution difficult. Of particular interest is the increased step reduction in crack growth rates of the equibiaxial 200MPa stress range. Under the stress intensity range regime of Figure 94 and Figure 95 this step reduction was such that crack growth rates were still slightly higher than the unibiaxial. However, with the plastic zone parameter, they are now slightly lower.

Introduction of thermal shock to isothermal crack growth rates is made in Figure 117. Here the von Mises plastic zone parameter is used with only the maximum normal plastic zone of Figure 113. The importance of a complete description of the plastic zone state is immediately obvious. The single edge shock with a mean of 120MPa shows reducing crack growth rates with increasing magnitude of the plastic zone parameter. Introduction of the minimum normal plastic zone produces Figure 118 where the distribution is changed dramatically.

Problematic forms, however, still exist in the distribution. At high crack growth rates in both single-edge loading scenarios the plastic zone parameter is seen to reverse in magnitude. These reversing crack growth rates take place at short crack lengths when the severity of thermal shock loading is greatest. Although the plastic zone is a direct function of the stress intensity the plastic zone radius responds to increases in crack length at a slower rate than the stress intensity. Consequently, under the conditions of high crack growth rates from high stressing on small cracks the plastic zone is restrained more by the shortness of the crack than increased by the magnitude of the loading. As a result crack growth data will encounter reversing at small cracks with high growth rates. This would not be expected in the isothermal biaxial loading data since no oscillation of the load takes place over the given range of crack lengths.

6.4.5.2 – STRESS INTENSITY FACTOR

Returning to the stress intensity factor, results become far more legible. Still applied over a narrow range in comparison to the isothermal data the results do demonstrate an excellent correlation to the isothermal data.

Figure 119 shows the elastic-plastic crack growth rates in relation to the linear-elastic isothermal biaxial data. Though the double-edge mean load data still fall well outside the region of the majority of data, much of the growth rates for thermal shock fall within the low load biaxial region. If the effects of plasticity are introduced to the

isothermal growth rates the correlation is improved further. Shown in Figure 120 the majority of thermal shock growth rates now correlate excellently within the isothermal growth rates, specifically, between the equibiaxial and unibiaxial at a stress range of 200MPa.

6.5 – Quantification of Thermal Loading

6.5.1 – *h* Determination

The FD&T method described previously has been used to determine the effect of start, end and ambient temperatures and time period on h and σ_{MAX}. A series of one dimensional finite difference temperature dependent thermal models were conducted and are listed in Table 25.

The determination for a function of h is found from the normalised thermal gradient function, eqn. 79. When these temperatures are normalised in this fashion an excellent power law relationship is found between this explicit thermal cycle definition and their required h value. This is illustrated in Figure 121.

The effect of time period is shown by a vertical shift of the results inversely proportional to the change in time period, i.e., the longer the time period the lower the required h value. Figure 121 also demonstrates the increased difficulty of forced convection cooling to ambient conditions as predicted previously by eqn. 80. The power law shows good correlation for all shocks with a T_{FACTOR} less than around 0.75, however, due to lack of a zero boundary condition for no shock at $T_{FAC} = 1$ the approximation should be avoided for thermal cycles with small ΔT or T_{FACTOR} values greater than 0.8. This can easily be overcome by replacing eqn. 79 with eqn. 91, which would introduce the condition of $T_{FACTOR} = 1$ at $T_{END} = T_A$.

$$T_{FACTOR} = \left[1 - \frac{T_{END} - T_A}{T_{INITIAL} - T_A}\right]$$

eqn. 91

However, further terms in the final description are required in order to make convergence. This leads to bulky handling of constants not conducive to a complete solution. It is also clear from Figure 121 that the 3, 4 and 5 seconds time periods are very similar in result. Though having a marked effect on stress as will be illustrated in the following section, little change is found over the three time periods. So much so, that it is only relevant to illustrate the bounds of the conducted models, 3 and 5 second time periods. This is reflected in the final analysis shown by eqn. 92 below, describing the relationship of h to the normalised thermal gradient, T_{FACTOR}.

$$h = \frac{5885}{\sqrt{t_P}}\left[\left(\frac{T_{END} - T_A}{T_{INITIAL} - T_A}\right)^{-1.199 t_P^{3.25E-3}} - 0.86\right]$$

eqn. 92

The time period must occur in both the coefficient and exponent, however, its presence places a dual effect on the function. From within the exponent, time period strongly holds the exponents' magnitude to low values, however, as time period increases the exponent increases and consequently the magnitude of h. This is in contradiction to that expected, increasing time periods should required a lower heat flux for a given T_{FACTOR}. The effect of this apparent discrepancy is, however, offset by the time period within the coefficient. Here, an inverse square law is found demonstrating the non-linear rise of h with reducing time period. Not only does the model predict infinite heat flux for an instant reduction in temperature, but also predicts the rise will be significantly non-linear.

Once the h value is determined according to the explicit description of the thermal load, it can be used to obtain the characterising maximum stress. Results are described in the following section.

6.5.2 – Maximum Elastic Stress

Corresponding stresses were calculated using thermomechanical properties of AISI 316 stainless steel specified at the maximum temperature of the shock.

Though a realistic stress analysis of thermal shock is more complex, the FD&T thermoelastic theory is adequate to provide some indication of the loading severity of a thermal shock cycle. Its relative simplicity is also convenient for evaluation of the large number of models conducted.

When maximum stress is plotted against the h value, presented in Figure 122, a very distinctive non-linear relationship is found supporting the non-linear behaviour of the h value to T_{FACTOR}. Each point on the plot represents the shocked edge maximum stress of a single model and illustrates the three distinctive factors of thermal loading; thermal gradient, ΔT, time period, t_P, and proximity to ambient conditions, T_{END}-T_A. Each line represents a specific initial temperature and is separated into groups of ΔT (the greater ΔT the higher the stress). Each group consists of the three different time periods, 3, 4 and 5s.

It can easily be seen that the effect of time period even for small changes can be large for small shocks whilst for large shocks it can become negligible. It is important to note at this point that although initial temperature elastic modulus values were used, there was only 5-10% change in modulus over these temperatures. Consequently the effect of initial temperature is clearly as important as the thermal gradient.

The effect of proximity to ambient conditions on a thermal cycle can be more clearly seen in Table 25. With T_A fixed, for any given t_P and ΔT the resulting maximum stress can be very similar (an example is highlighted in bold within the table). However, the required h value to generate the thermal cycle can be two or three times greater for the lower $T_{INITIAL}$. This is due to the increased proximity of T_{END} to ambient conditions. In natural cooling the rate of cooling is drastically reduced when approaching ambient conditions, consequently greater effort is required when forcing temperatures to near ambient conditions.

This has costly implications in thermal fatigue rig design. To induce high thermal gradients rapidly, expensive thermal apparatus is required. The economical advantages of using relatively low temperatures is offset by the increased forced convection required to gain large thermal gradients with end temperatures close to ambient conditions.

The overall relationship shown in Figure 122 is very similar to that of a Ramberg-Osgood style stress/strain response; however, such a relationship requires iteration procedures to solve. A good approximation can be determined by a simple log function. The entire set of data can be conveniently described by a single function given by eqn. 93 where K is a constant and $E\alpha$ is the product of the linear-elastic modulus and the thermal expansion coefficient at the maximum temperature, or initial conditions of thermal downshock.

$$\sigma_{MAX} = \left| T_{INITIAL} \left[K \log(h) - E\alpha \right] \right| \qquad K = 1.123E+6$$

eqn. 93

Thus, it has now become possible to quantify the stress state of a component subject to thermal shock loading by an explicit and practical description of the thermal downshock loading present.

7 – Discussion

7.1 – Thermal Analysis

7.1.1 – Thermal Distribution

Three methods have been used to determine a thermal shock temperature distribution: theoretical, finite difference and finite element. Only edge cooling has been modelled, neglecting the global cooling in experimental data provided by Hasan [124], caused by turbulence around the specimen from the air jets. This was not considered as a quantified event and an idealised thermal shock scenario was maintained in the models. This allowed a welcome simplicity to the models whilst also providing a pessimistic evaluation of thermal shock.

Both fixed and temperature dependant thermal properties have been used in the finite approximations and compared to test results, as illustrated by Figures 49 through 53. Fluctuations in fixed and temperature dependant models were small, due to the relatively small changes in thermal properties, i.e. specific heat, density and thermal conductivity. Consequently, their relative difference to experimental data was small.

A good approximation to the experimental data was obtained at near shocked edge regions whilst the global cooling effect reduced mid-region and opposite-edge region temperatures. For the experimental set-up this would reduce the overall thermal gradient and so reduces stresses. Under single edge loading, however, the reduction is not great and the model stresses show a good comparison to those determined by the experimental thermal distribution. This is a result of the global cooling encountered in the experimental study resulting in an end of cycle temperature at the mid-region of the specimen not far removed from the idealised model.

Very little difference is found in the T_{FACTOR}, as described in eqn. 79, for the experimental and idealised model. For the 650-310°C in 5s thermal shock the T_{FACTOR} from eqn. 79 is 0.461 for the model and approximately 0.475 for the experimental results, see Figure 49 where $T_{INITIAL}$ is taken as the mid-region temperature at the end of the time period. This represents an increase of T_{FACTOR} and hence a loss of loading severity of only 3%. This is reflected in the corresponding stresses, illustrated in Figure 123. For the most extreme shock conducted, 625-225°C in 3s, stresses for both

experimental and thermal distributions show little difference for a single-edge shock scenario. Consequently the single edge models can be used in relation with crack growth models taken from the previous experimental study to demonstrate a correlation to isothermal models.

7.1.2 – Quantification of Thermal Distribution

The attempt to provide an explicit definition of the thermal shock loading severity by the explicit parameters of the thermal gradient, ΔT, and time period, t_P have been very successful. However, the method of initial cooling rate has been shown to be an unrefinable quantity in numeric analyses. Further to this, since the theoretical cooling rate is an infinite series summation, deriving an initial gradient function by a derivative is arduous at best. This flaw is also reflected in the linear-average cooling rate, $\Delta T/t_P$, correlated to the maximum tensile stress in the downshock cycle as shown in Figure 124. A fairly coherent but widespread set of data is obtained, with tiers and dual points for a given linear average. The tiers describe time period, the smaller the time period the higher the stress and the higher the tier the higher the thermal gradient. Dual points are a result of the same thermal gradient and time period but with a different initial condition, the lower the initial condition the higher the stress. Though little difference separates the dual points at low linear-averages, the difference is directly proportionate to the increasing linear-average. The consequence, however, is that of a very crude description of the thermal shock loading severity. Any explicit quantification of thermal loading must account not only for thermal gradient and time period, but also for initial conditions or more accurately the proximity of initial conditions to ambient conditions. This is quantified by eqn. 94, where A, B and C are constants equal to 5885, -1.199 and 3.25E-3 respectively.

$$h = \frac{A}{\sqrt{t_P}}\left[\left(\frac{T_{END} - T_A}{T_{INITIAL} - T_A}\right)^{B \cdot t_P^C} - 0.86\right]$$

eqn. 94

The function shows itself to be quite capable of predicting a range of time periods over T_{FACTOR} values between 0.2 and 0.8 and is illustrated in Figure 125. Time periods of 0.1 to 100 seconds are indicated with the finite difference model solution

for comparison. Since this data is taken from the finite difference model the solution is not exact. As shown previously in Figure 54 the solution is around 5-10% more pessimistic than the FE and theoretical solutions, therefore constants A, B and C are solution method dependent. Consequently they cannot be related to the likely thermal transient parameters of thermal conductivity, density and specific heat and length. However, since the FD solution has been verified both by FE and theoretical thermal distributions, only the magnitudes of A, B and C will be effected and not the form of the equation. Consequently the likely form of the three constants will be related to the thermal resistance, R, as

$$A,B,C = f(R) = f\left(\frac{L}{kA}\right)$$

eqn. 95

In this case L is the width of the specimen, k the thermal conductivity and A the heat flow sectional area.

7.1.3 – Double-Edge Scenario

In the double-edge scenario, the cooling at the mid-region is much greater than in a single edge scenario, generating a lower overall thermal gradient and this is reflected by reduced stresses in this region, as illustrated by Figure 126. Using T_{FACTOR} again as a quantification of the difference in thermal distribution, the model determines a T_{FACTOR} of 0.473 and experimental is determined at 0.549, see Figure 50. This demonstrates the previous supposition of section 6.2.2.2 that heat transfer coefficients superimpose at the mid-region of the experimental set-up. This can be expressed functionally for single edge and double edge shocks as follows.

$$h_S = h - f(x^n) \qquad 0 < x < W \qquad \textit{single edge}$$

$$h_D \approx h_S \qquad 0 < x < C\frac{W}{2} \qquad \textit{double edge}$$

$$h_D \approx 2h_S \qquad C\frac{W}{2} < x < \left[W - C\frac{W}{2}\right] \qquad \textit{double edge}$$

Under single edge loading the heat transfer coefficient, h_S, is a variable across the width surface in the experimental set-up. As the distance through the width increases, h_S will reduce from h at the shocked edge directly as a function of distance.

When double edge is introduced, h_S will not be greatly changed at the near edge region defined as a fraction, C, of the half width $W/2$, as long as C is small, i.e. in the vicinity where thermal gradient approaches zero. In the mid-region, however, it is proposed that h_S of separate single edge scenarios combine to create a higher value, h_D.

This does not, however, increase the severity of the thermal shock loading, but rather reduces it. By increasing h to h_D in the mid region the temperature of the bulk material is reduced. Consequently, near edge thermal gradients are reduced resulting in lower overall stresses at any given time.

7.2 – Stress-Strain Analysis

7.2.1 – Biaxiality

Localisation of the thermally shocked region under a quarter of full-length exposure creates significant biaxiality. This is an effect which to date has only been addressed by Bass *et al* [77] for the fracture of surface flaws in RPV's for PTS. Since no mechanical loads are applied in generation of thermal shock stresses, and component stresses are both tensile and compressive, it is difficult to quantify the biaxiality. At positions where stress crosses from tensile to compressive the biaxiality defined as a ratio of lateral stress to normal stress approaches infinity. This emphasises the importance of an accurate correlation of isothermal mechanical data to thermal shock data.

Descriptions of biaxiality have been presented: specifically a linear and quadratic relationship. Since these functions are variable across the shocked specimen width it is necessary to place bounding limits on the biaxiality by maximum and minimum values. A maximum of 1 must be considered for both linear and quadratic functions. For thermal downshock, at an optimum localisation of 16mm over a 212 mm long specimen, a minimum biaxiality for the linear and quadratic functions is around 0.35 and 0.65 respectively. Isothermal crack growth rates used here are determined from loads of a minimum biaxiality of 0.55 and 0.84 for the linear and quadratic functions respectively.

Therefore, the quadratic function produces higher biaxiality ratios than the linear. Meaning it is less sensitive to changes in biaxiality, using the linear function

would create a larger spread of loading cases. Since only the bounding biaxial limits are required for any given thermal shock scenario the anomalous singular values created by the linear function, see Figure 73, can be ignored. Consequently the linear function of biaxiality can be used to determine the lower biaxial limit encountered in the given thermal shock scenario. Its larger spread of values, over the quadratic, making it less narrow in selection of possible load cases. Once this is determined the appropriate magnitude of the isothermal load can be addressed and coupled with the biaxiality used to define the isothermal crack growth tests cases bounding the given thermal shock scenario.

7.2.2 – Effect of Plasticity

Introduction of plasticity creates a reduction in the high tensile elastic-stresses at the surface in direct relation to αT passing an effective yield strain. This is expected and shown by Hasan, [64]. However, introducing plasticity to eqn. 36 is an over simplification in that the equation does not account for normal strains. Whilst not affecting a linear-elastic analysis, the plasticity induced by these strains is significant enough to increase the general stress distribution, generating a longer tensile region ahead of the shocked edge. This is a result of the inherent biaxiality present in plasticity. With high biaxiality a one-dimensional analysis such as eqn. 36, is inadequate since it is unable to account for plastic incompressibility. Normal strains responding to large lateral strains induced by the thermal shock profile are created as an integral effect. Therefore, when a one-dimensional analysis is conducted with plasticity the complete strain response is not modelled and therefore underestimated. When a full Ramberg-Osgood elastic-plastic material model is used the under estimation of stress by eqn. 36 is marked both in magnitude and shape, this is illustrated in Figure 83.

7.2.3 – Bending Effects

Increased stresses observed under symmetrical double-edge thermal shock are a result of the shock applied to one edge constraining free bending of the opposite edge. When a single edge shock takes place the rate of heat flow from the surface carried

away by the cooling film is such that the bulk material does not cool. When held in expansion this bulk material throws the shocked surface, attempting to contract, into tension. However, some contraction of the cooled edge does take place and bending into the cooled edge occurs. When this is counteracted by an equal thermal shock on the opposite edge, contraction forces cancel each other out and bending is restrained. This induces further tensile stresses adding to the single-edge stress produced by the same thermal distribution. This is demonstrated by normal strains distributing horizontally in the absence of bending.

Further to this, global cooling also reduces bending. Air jet cooling tests conducted by Hasan [124] cool the mid-region of the specimen as well as the shocked surface, albeit at a much lower rate. Nevertheless thermal gradients will be reduced and consequently the overall stress distribution will fall. This will reduce the absolute difference between tensile stress at either edge, consequently reducing bending. If bending is reduced, the amount of crack tip opening is increased and crack tip closure at near shocked-edge crack lengths reduces, consequently crack growth rates will increase. This is described in more detail in a later section.

The amount of crack tip closure taking place is a summation of the opposite effects of tensile stress, just behind the crack tip and compressive loading of the cracked face by bending. This summation is in the normal tensile stress region, hence any crack ahead of the tensile region is under compressive bending, illustrated by Figure 127. However, at the far side of the specimen, stresses again pass into a tensile state as the roles of thermal membrane and bending stress are reversed.

The consequence of this to crack closure by oxidation effects, at newly formed crack surfaces, is that oxidation will have a rapidly increasing effect on decelerating crack growth rates. As the tensile stress rapidly falls at the edge, the force available just behind a crack tip falls with it. This assumes the crack tip stays on the same symmetrical path until total crack closure or catastrophic failure occurs. In the event of crack spalling due to crack tip blunting tensile force can still be maintained around the crack tip as it rotates away from the compressive bending zone.

7.2.4 – Quantification of Thermal Stress

7.2.4.1 – FULL EDGE SHOCK

Further to quantifying the thermal distribution it is possible to do the same for the thermal stresses. Section 6.5.2 shows in more detail the derivation of eqn. 96.

$$\sigma_{MAX} = \left| T_{MAX} \left[K \log(h) - E\alpha \right] \right| \qquad K = 1.123E+6$$

eqn. 96

Once the heat transfer coefficient, h, has been determined from eqn. 94, then eqn. 96 can be used to determine the maximum elastic tensile stress at the shocked-edge. These stresses are determined by a linear-elastic analysis using the FD&T method with a 1D thermal distribution. As with the thermal quantification, constants can not be fully described in terms of material properties since the analysis uses an over estimation of the heat transfer coefficient. It is shown, however, that the maximum tensile stresses can be accurately described by a simple dual equation method.

To account for localisation, eqn. 96 must be factored by a general shape function. However, illustrated in Figure 69, the shape function is not straightforward, and requires either a bulky two form equation or the normalised function to be read from a chart as illustrated in eqn. 97 and Figure 128.

$$\textit{Shape Function} = f\left(\left[\frac{l}{L} \right], W^2 \right) \qquad \text{for } 0 < \frac{l}{L} < 0.25$$

$$\textit{Shape Function} = 1 \qquad \text{for } 0.25 <= \frac{l}{L} < 1$$

$$\textit{where } l = \textit{localised height} \quad L = \textit{specimen height}$$

eqn. 97

Although such a method is not as elegant as a single function proposed by eqn. 96, it has the advantage of dependence only on localisation and specimen width. Since the reduction in stress due to localisation also reduces bending, the specimen width will have a further effect on this shape function. From simple bending theory the likely form will be of a second order width term in the shape function.

7.3 – Fracture Analysis

7.3.1 – Plastic Zone Parameter

Though the plastic zone parameter demonstrates the behaviour of a biaxially loaded crack tip well, it does not perform well under interrogation of thermal shock crack growth rates. A possible explanation to this is illustrated in Figure 129 and Figure 130 where the increasing stress intensity factor is compared to the increasing plastic zone size under increasing crack length and loading respectively. It is clear that both stress intensity and plastic zone methods are very different. The plastic zone response to increasing crack length and applied load is always slower than that of the stress intensity factor. The primary difference on crack tip loading between isothermal biaxiality and thermal shock loading conditions is that under thermal shock loading the loads experienced by the crack tip are changing in accordance with the thermal stress distribution.

Figure 129 demonstrates a linear response of increasing plastic zone with increasing crack length and a non-linear increase in stress intensity. Therefore, if the crack length is doubled at a given load, the plastic zone size will double but the stress intensity will rise by only $x\sqrt{2}$. The opposite is illustrated for changing loads at a given crack length by Figure 130. Consequently the plastic zone is less sensitive to changing loads than to changing crack lengths. Therefore, under thermal shock with changing loads for increasing crack lengths, the plastic zone will be repressed at small crack lengths and exaggerated at larger crack lengths. This is a relative effect on variable loads with increasing crack lengths and is consequently not observed under isothermal biaxial conditions where loads are applied at a constant magnitude.

The end result of this is a uniform response of increasing crack lengths to isothermal crack growth rates making them able to be correlated using the plastic zone parameter, shown by Figure 116. Thermal shock crack growth rates however, show high growth rates from high loads at small crack lengths and low growth rates from low loads at longer crack lengths, indicated by Figure 117. As the plastic zone is more sensitive to crack length than load, the plastic zone parameter will inevitably be small at small crack lengths even though the load is high. Further more, at longer crack lengths, the plastic zone parameter will tend towards higher values even under the presence of low loads.

7.3.2 – Isothermal-Thermal Shock Crack Growth Correlation

Using both plastic zone parameter and stress intensity factor the same behaviour of isothermal biaxiality is observed. At low loads, unibiaxial loading produces higher crack growth rates than equibiaxial loading. However, at high loads, the opposite is observed. This is considered to be a result of crack tip blunting due to increased plasticity ahead of the crack tip. At high loads, plastic strain around the crack tip will be disproportionately high resulting in greater shear decohesion and so greater blunting. Under the influence of a high compressive lateral stress, lateral strains will cause greater blunting under equibiaxial loading than unibiaxial, since the lateral component is equally high. Further intermediate load tests are required to confirm this response.

An excellent correlation of isothermal to thermal shock crack growth rates is made by the high isothermal loads of $\Delta\sigma$ = 200MPa. Crack lengths for both models exist in very similar ranges, 3-10mm for isothermal data and 1-7mm for thermal shock data, making the need for a function relating crack lengths between the two models unnecessary.

Using this data, a Paris type equation can be developed to describe the bounding isothermal limits on the thermal shock crack growth rates.

$$\frac{da}{dN} = C\left[1 - A\frac{\Delta\sigma_T + \sigma_{T\text{-}MEAN}}{\Delta\sigma_N}\right](\Delta K_I)^m$$

eqn. 98

Presented in eqn. 98, a standard Paris type equation is modified to include the additional effect of σ_T, where $C = 4.5x10^{-9}$, $m = 4$ and $A = 0.589$. Using the σ_T value as a description of the biaxial loading conducted at an R-ratio of 0.1 the total sinusoidal load form used in the cruciform models can be described. In the case of the isothermal cruciform models conducted, the σ_T value is equal to the total stress range of the applied normal load cycle. The value of $\sigma_{T\text{-}MEAN}$ is also zero, this occurs under these loading conditions when the lateral dead load is held at the mean of the normal load, σ_N. If the lateral load was not held at this value, i.e. in near uniaxial loading, σ_T would no longer oscillate about zero. It is proposed that this would increase crack growth rates and is consequently positioned in the function as an increasing addition to the stress range.

A condition of zero σ_T is set for equibiaxial loading. Under unibiaxial loading, the lateral load σ_L, equates to the mean of the applied normal load, σ_N, consequently the value of $\sigma_{T\text{-}MEAN}$ equates to zero.

Further work is needed to clarify its range of use and relationship of applied biaxial loads to thermal shock loading severity. By using the correlation of eqn. 98 thermal shock crack growth rates can be estimated from isothermal biaxial test data.

The thermal shock and isothermal cruciform models studied show a distinct correlation in crack growth rates, as illustrated in Figure 120. The figure demonstrates the ability of three different levels of thermal shock, 625-225°C in 3s single-edge and 612-300°C in 5s double edge, to be bound at its upper and lower limits by the 200MPa stress range of the isothermal cruciform models. Also included are the effects of end load on both scenarios. In the single edge scenario, Figure 120 still demonstrates a good correlation. Interestingly, the rate of crack growth has been reduced into a compact region of shallower gradient. Consequently, the effects of mean load on thermal shock do not appear to be simply an accelerating factor, rather one of levelling to a smaller growth rate. This can be explained by the summation of a constant mechanical load, of magnitude similar to the maximum compressive normal stress, to the stress by thermal shock. In the mid-region where the compressive stress is alleviated by the applied load, crack growth rates will be increased by the reduction in compressive stress. However, only small tensile stresses in this region may be present and consequently a crack tip parameter such as stress intensity range, does not undergo large reductions, instead it continues to slowly increase producing crack growths of a narrow band as illustrated in Figure 120.

The double-edge dead load data shows no correlation to the isothermal data. Though few data are presented it must still be considered as a possible genuine response. As with single edge data, the response of a narrow crack growth rate band over a relatively large stress intensity range is present. However, illustrated by Figure 88, the difference in compressive normal stress is not so different from that of the single-edge whilst the tensile stresses are significantly different. With a large applied dead load of 120MPa the resulting amplification on the maximum tensile stress is 50%. This would create a much larger tensile region in the specimen and enlarge the tensile load in the mid-region. Consequently, very high crack growth rates would be a

result. This could therefore mean the data of 612-300°C in 5s is genuinely created from the excessive dead load of 120MPa.

It is interesting to note from Figure 88 that the maximum tensile stress at the surface for high localisation of the single edge scenario is equal to the stress range of the isothermal cruciform tests, 200MPa. The localisation of the thermal shock test data is 16mm in a 212mm long specimen. According to Figure 128 this would place the maximum elastic-plastic thermal stress in the region of 250MPa, where, with an R-ratio of 0.1, the maximum normal load in the isothermal models is 222MPa. For double edge at 612-300°C in 5s this would be reduced to 200-250MPa. This shows that the maximum tensile stress of elastic-plastic thermal shock to be approximately equal to that of the correlating stress range of isothermal data at an R-ratio of 0.1.

Consequently, the preliminary proposal can be made that isothermal biaxial crack growth rates can be used under bounding biaxial conditions to correlate the thermal shock growth rates when loaded to the maximum normal EP thermal shock stress.

Once this value is known the bounding isothermal crack growth rates can be calculated from eqn. 98, using σ_T and the biaxial ratio determined from Figure 73. Thermal shock crack growth rates can then be estimated to lie within these lines. Since thermal shock stress intensity ranges, or other parameters such as plastic zone, are in a narrow band range essentially centred within the isothermal data it is reasonably simple to approximate a stress intensity range. This can be done by a straightforward fracture model or determined from a Greens function as previously presented by eqn. 38 [124]. Whichever route is taken, it is a cheaper and less arduous method to estimate the effects of thermal shock on crack growth rates.

8 – Conclusions

8.1 - Thermal Analysis

A comparison is made of theoretical single-edge and double-edge shock thermal distributions to experimental results of a previous study [124]. Both theoretical and numerical solutions are determined, with numerical solutions consisting of both finite difference and finite element analyses at fixed and temperature dependant thermal properties for AISI 316.

It is found that the change in thermal distribution due to temperature dependence is only slight. This is a result of only marginal changes in thermal properties over the large temperature ranges used. Comparison between numerical and theoretical thermal distributions is excellent, as should be the case, since both solutions are based on the same first principles of conduction and convection.

Both solutions are ideal in that comparison to air jet convection tests, they do not account for the diverging stream of turbulent air on either side of the specimen. In near shocked edge regions, approximately one third of the specimen width, numerical and theoretical models compare very well to the air jet experimental data. However, in mid and opposite-edge regions, models show a constant temperature of initial conditions. This does not compare well to the experimental data, which shows a drop of up to 5% of initial conditions. This is marked even further under double-edge shock where the superposition of heat transfer coefficients generates higher levels of cooling than single-edge in the experimental data, resulting in up to a 15% drop in mid-region temperatures from initial conditions. This is demonstrated in stress analyses to have little significant difference.

It is also demonstrated that a wide range of heat transfer coefficients can be estimated to within 3% from the explicitly defining characteristics of thermal shock for AISI 316 in air; initial conditions, shocked surface end temperature, ambient temperature and time period. Made in relation to a thermal gradient factor, T_{FACTOR}, the relationship holds for a wide variety of thermal gradients covering factors of 0.15 to 0.8.

The effect of localisation on a 2D thermal distribution model are shown to be essentially non existent at the symmetry plane to that of a 1D thermal distribution model.

8.2 – Stress-Strain Analysis

The effects of lower mid-range temperatures, due to diverging airflow under experimental conditions, are shown to be negligible on single edge stresses. However, under double-edge shock the effects begin to be more pronounce. This is due simply to the greater reduction in temperature in these regions from superposition of heat transfer coefficients. Therefore, stresses could be reduced in single-edge shocks over longer time periods, where the thermal gradient has time to conduct further into the mid region. Consequently, at longer time periods, models will require to account for the effects of turbulent airflow other than that at the shocked edge. For the models presented here, however, the effect is negligible.

Localisation of the thermally shocked region has two significant effects. The normal tensile stress at the symmetry plane is found to decrease in the region of 50%-60%. The introduction of biaxiality also occurs due to the generation of a compressive lateral stress. A linear biaxial function is felt best to describe this level of biaxiality. The quadratic function, proposed to avoid divide by zero events, compresses the range of biaxiality to approximately half that of the linear function making it less sensitive to changes in biaxiality. The biaxial range of the thermal shock models is shown to be 0.35 to 1, with 0.35 representing the position of maximum compressive lateral stress. This effect however, appears to be short lived, once the shocked region is greater than approximately 25% of the total surface, normal tensile stresses are maximised. The experimental shock tests are conducted at a localisation of approximately 10%, meaning stresses are lower than a 1D thermal analysis would suggest. This also coincides with a maximum compressive lateral stress state.

Under double-edge shock, stresses are demonstrated both by the finite element method and by the FD&T method to be higher than those of single-edge stresses. This is a result of each shocked edge restraining compressive bending of the opposite edge. As a result the single edge compressive bending stress is thrown into tension superimposing itself on the tensile stress caused by thermal shock at that surface.

Reduced crack tip closure is a likely consequence of this restrained bending and hence higher crack growth rates result.

Quantification of maximum normal thermal stress is described by a simple log function of heat transfer coefficient and elastic thermomechanical properties E and α. Used in conjunction with the estimation of heat transfer coefficient a very quick and accurate estimation of maximum normal elastic stress can be made for single-edge shock. Since the increase in stress for double-edge shock is a function of the single edge shock bending stress it will be universal and not dependent on material properties or level of shock. Therefore, the single-edge estimation can simply be factored to obtain the double-edge stress.

8.3 – Correlation of Crack Growth Rates

The plastic zone parameter derived to introduce a description of biaxiality on the crack tip is shown to illustrate the nature of both isothermal biaxial and thermal downshock loading. However, when introduced to crack growth rates it is clear that the plastic zone parameter is not sufficient. A result of the plastic zones tendency to be governed more by the length of crack than magnitude of loading shows a broader ranging flaw in the use of plastic zone on loads which oscillate over increasing crack lengths.

As a result a mode I stress intensity range function is determined with a linear addition of biaxial loading. This allows the nature of loading at a crack tip to be quantified outside the effects of biaxiality, with an additional effect by biaxiality accounted for based on the level of non-singular σ_T and biaxial ratio.

Also proposed, is the introduction of a mean value of σ_T. Using a sinusoidal normal load with a lateral dead load equal to the mean of the normal load, creates unibiaxial conditions and values of σ_T oscillating around zero. Should the lateral dead load be a value other than the mean of the normal load a mean σ_T will result. It is proposed that this, with increasing negative values of σ_T, as found by Kfouri and Miller [33] will increase crack growth rates and is included on this basis.

It is observed that for the models conducted, the maximum tensile normal stress of the thermal shock scenarios using elastic-plastic properties are very similar to those of the peak isothermal biaxial loading. It is therefore proposed that the peak thermal

shock stress determined with elastic-plastic properties is approximately equal to that of the bounding isothermal biaxial peak load with an R-ratio of 0.1.

8.4 – Proposed Method of Correlation

It is now possible to propose a complete method of correlating thermal shock crack growth rates with isothermal biaxial bounding crack growth rates.

1. Estimate h from eqn. 94 and explicit definitions of thermal shock cycle.
2. Use estimation of h in eqn. 96 and Figure 126 to estimate thermal stress.
3. Equate thermal stress to peak isothermal load at R = 0.1.

This method, however, does not account for elastic-plastic properties. This requires further work into determining the exact form of eqn. 96 for an elastic-plastic material response. Since it is shown that eqn. 36 inadequately represents plasticity, and thermal stresses dependant on localisation, full finite element models would be required to determine eqn. 96 for plasticity. The form of the equation will be very much like that of the elastic version but shallower for high h values.

Once achieved, however, it will produce a description of an isothermal loading stress range. When coupled with biaxial ratios bounding thermal shock, i.e. 0.35 and 1, the bounding crack growth models can be calculated.

Consequently, an estimation of thermal downshock crack growth can be made using a simple analysis of the explicit conditions defining the thermal downshock cycle, i.e. initial conditions, end temperature at the shocked surface, ambient temperature and time period.

9 – Recommendations

The work presented here demonstrates that a correlation exists between thermal shock and isothermal biaxial crack growth rates. The crack growth rates used however, do not cover a broad range of loading scenarios and as such further study is needed to confirm the conclusions made.

This study must take the form of the experimental work in both isothermal and thermal shock loading. In order to study the manner in which biaxiality changes with increasing loads it is necessary to conduct a range of test on the cruciform specimen through the elastic and elastic-plastic range. This would mean tests with stress ranges from 20MPa to 200MPa to encompass the complete behaviour under a full order of magnitude change.

It will also be advisable to study particular levels of load, i.e. at either end of the scale, with various levels of biaxiality. It is suggested that equibiaxial, unibiaxial and biaxial levels be studied, where biaxial tests would be conducted at the minimum thermal shock biaxiality of 0.35. This would also introduce the effect of mean σ_T in to the results.

This should be done in conjunction with finite element models on thermal shock with elastic-plastic material properties. From these models an elastic-plastic thermal stress can be estimated as presented here for elastic stress. This will guide the direction of choosing the specific magnitudes of isothermal stress ranges to use in tests. Once this magnitude is determined, a spread of ±50% should be applied to better define the relationship between peak elastic-plastic thermal stress and peak isothermal loading, here proposed to be approximately equal.

Double-edge thermal shock models should also be studied with complete description of the heat transfer coefficient on all convecting surfaces. A numerical 3D finite difference method similar to the 2D method already described could be applied to known experimental 2D thermal data for such types of shocks. Since the heat transfer is transient and surface temperatures can be determined from pyrometer readings, only initial conditions and the time dependent surface temperatures would be needed to determine heat transfer coefficients local to each node of the model. This would provide a complete description of biaxiality and its relationship between thermal shock and isothermal biaxial tests.

References

[1] George E. Dieter, Mechanical Metallurgy, McGraw Hill Metric Ed, 1986, pp396.

[2] B. Tomkins, Fatigue Crack Propagation - An Analysis, Philosophical Magazine, vol. 18, 1968, pp. 1041-66.

[3] G. R. Irwin, Analysis of Stresses and Strains Near the End of a Crack Traversing a Plate, Journal of Applied Mechanics, vol. 24, pp. 361-364, 1957.

[4] F. Erdogan, Stress Intensity Factors, Journal of Applied Mechanics, vol. 50, pp. 992-1002, 1983.

[5] D. P. Rooke, D. J. Cartwright, Compendium of Stress Intensity Factors, MOD 1974.

[6] T. S. Gross, ASM Handbook, Vol.19, Fatigue & Fracture, 1996, pp42.

[7] ASM Metal Handbook, 9th Ed., vol. 9.

[8] ASM Metal Handbook, 8th Ed., vol. 8.

[9] K. J. Miller, The Fracture of Engineering Materials, Dept. Mech. Eng. University of Sheffield.

[10] M. F. Ashby, A First Report on Deformation-Mechanism Maps, Acta Metall. 1972, 20, 887-892.

[11] D. J. Marsh, A Thermal Shock Fatigue Study of Type 304 & 316 Stainless Steels, Fatigue of Engineering Materials and Structures, Vol.4, No.2, 1981, pp179-195.

[12] D. J. Marsh, F. D. W. Charlesworth, The Determination and Interpretation of Thermally Promoted Crack Initiation and Crack Growth Data and its Correlation with Current Uniaxial Design Data, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown Eds., 1985, pp700-719.

[13] J. R. Rice, A path Independent Integral and the Approximate Analysis of Strain Concentration By Notches, Journal of Applied Mechanics, June 1968, pp. 379-386.

[14] J. R. Rice, Elastic-Plastic Fracture Mechanics, ASME Applied Mechanics Division, vol. 19, pp. 23-53, 1976.

[15] J. R. Rice, Two General Integrals of Singular Crack Tip Deformation Fields, Journal of Elasticity,vol. 20, pp. 131-142, 1988.

[16] T. K. Hellen, The Calculation of Stress Intensity Factors for Combined Tensile and Shear Loading, W. S. Blackburn, International Journal of Fracture, vol. 11, no. 4, pp. 605-617, 1975.

[17] W. Chen, K. Chen, On the Study of Mixed Mode Thermal Fracture Using Modified J_K Integrals, International Journal of Fracture, vol. 17, pp. R99-103, 1981.

[18] Y. Lambert, P. Saillard, C. Bathias, Application of the J Concept to Fatigue Crack Growth in Large Scale Yielding, Fracture Mechanics: 19th Symposium, ASTM STP 969, T. A. Curse, Ed., Philadelphia 1988, pp. 318-329.

[19] L. Banks-Sills, Y. Volpert, Application of the Cyclic J-Integral to Fatigue Crack Propagation of Al 2024-T351, Engineering Fracture Mechanics, vol. 40, no. 2, 1991, pp. 355-370.

[20] J. Mao, Q. Hong, R. Wang, M. Hua, Damage Tolerance Analysis of Compressor Blade, International Gas Turbine & Aeroengine Congress & Exposition, 13-16 June 1994, ASME New York, pp. 1-4.

[21] A. Bobet, H. H. Einstein, Numerical Modelling of Fracture Coalescence in a Model Rock Material, International Journal of Fracture, Vol. 92, Iss. 3, 1999, pp. 221-252.

[22] R. R. Bhargava, K. Rajesh, Two Equal Symmetric Cicular Acr Cracks in an Elastic Medium – the Dugdale Approach, Journal of Engineering Mathematics, Vol. 34, Iss. 4, Nov. 1998, pp. 359-369.

[23] A. A. Shaniavski, E. F. Orlov, Model of Fatigue Crack Growth after an Overload in D16T Al-Alloy Subjected to Biaxial Cyclic Loads at Various R-Ratios, Vol. 20, Iss. 12, 1997, pp. 1719-1730.

[24] K. G. Farkhutdinov, F. A. Fairshin, Effect of Thermomechanical Treatment of 0.45 wt.% C Steel in the Superplastic Mode on Plastic Zone Parameters After Impact Fracture in the Ductile-Brittle Transition Temperature Range, Materials Science and Engineering, A190, 1995, pp. 117-123.

[25] H. Y. Ahmad, M. P. Clode, J. R Yates, Prediction of Fatigue Crack Growth in Notched Members, International Journal of Fatigue, Vol. 19, No. 10, pp. 703-712, 1997.

[26] K. J. Miller, A. P. Kfouri, An Elastic Plastic Finite Element Analysis of Crack Tip Fields Under Biaxial Loading Conditions, International Journal of Fracture, Vol. 10, No. 3, Sept. 1974, pp. 393-404.

[27] J. Eftis, N. Subramonian, H. Leibowitz, Crack Border Stress and Displacement Equations Revisited, Engineering Fracture Mechanics, 1977, pp. 189-210.

[28] J. Eftis, N. Subramonian, The Inclined Crack Under Biaxial Load, Engineering Fracture Mechaincs, 1978, Vol. 10, pp.43-67.

[29] K. J. Miller, A. P. Kfouri, A Comparison of Elastic Plastic Fracture Parameters in Biaxial Stress States, Elastic-Plastic Fracture, ASTM STP 668, J. D. Landes, J. A. Begley, G. A. Clarke, Eds., 1979, pp.214-228.

[30] A. P. Kfouri, K. J. Miller, The Effect of Load Biaxiality on the Fracture Toughness Parameters J and G^{Δ}, Fracture, 1977, ICF4, Waterloo, Canada, June 19-24, 1977.

[31] M. W. Brown, K. J. Miller, Mode I Fatigue Crack Growth Under Biaxial Stress at Room and Elevated Temperature, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown Eds., 1985, pp135-152.

[32] H. Kitagama, R. Yuuki, K. Tohgo, M. Tanabe, ΔK - Dependency of Fatigue Growth of Single and Mixed Mode Cracks Under Biaxial Stress, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown Eds., 1985, pp164-183.

[33] I. M. H. Chavrat, G. G. Garrett, The Development of a Closed-Loop, Servo-Hydraulic Test System for Direct Stress Monotonic and Cyclic Crack Propagation

Studies Under Biaxial Loading, Journal of Testing and Evaluation, JTEVA, Vol. 8, No. 1, Jan. 1980, pp. 9-17.

[34] D. Rhodes, J. C. Radon, Effect of Local stress Biaxiality on the Behaviour of Fatigue Crack Growth Test Specimen, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown, Eds. American Society for Testing and Materials, Philadelphia, 1985, pp. 153-163.

[35] A. P. Kfouri, K. S. Miller, Crack Separation Energy Release Rates for Inclined Cracks in a Biaxial Stress Field of an Elastic-Plastic Material, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown Eds., 1985, pp88-107.

[36] E. W. Smith, K. J. Pascoe, The Behaviour of Fatigue Cracks Subject to Applied Biaxial Stress: A Review of Experimental Evidence, Fatigue of Engineering Materials and Structures, Vol. 6, No. 3, pp. 201-224, 1983.

[37] D. A. Spera, What is Thermal Fatigue, Thermal Fatigue of Metals and Components, ASTM STP 612, D. A. Spera, D. F. Mowbray, Eds. 1976, pp3-9.

[38] E. R. de los Rios, F. A. Kandil, K. J. Miller, M. W. Brown, A Metallographic Study of Creep-Fatigue Behaviour in 316 Stainless Steel, Multiaxial Fatigue, ASTM STP 853, K. J. Miller and M. W. Brown, Eds., 1985, pp669-687.

[39] S. Nishino et al, Low Cycle Fatigue Crack Propagation Behaviour at 700°C Involving Oxidation and Coalescence of Cracks, Asian Pacific Conference on Fracture & Strength, JSME, 1993, pp157-161.

[40] T. Nakazawa et al, Study on Metallography of Low Cycle Creep Fatigue Fracture of Type 316 Stainless Steel, Low Cycle Fatigue and Elasto-Plastic Behaviour of Materials, 7-11 Sept. 1992, Berlin Germany, Elsevier Science Publ. Ltd., pp88-93.

[41] J. G. Park, D. Y. Lee, Creep Behaviour of AISI 316 Stainless Steel Above $0.5T_M$, Scripta Metallurgica et Materiala, Vol.29, 1993, pp595-598.

[42] Z. Shaikh, Effects of Temperature and Dwell Periods on Fatigue Behaviour of AISI 316 Stainless Steel, Proceedings of the 3rd International Symposium on Advanced Materials 1993, pp. 308-314.

[43] S. C. Kuiry, S. Seal, S. K. Bose, S. K. Roy, Effect of Surface Preparation on the High-Temperature Oxidation Behaviour of AISI 316 Stainless Steel, ISIJ International, vol. 34, no. 7, 1994, pp. 599-606.

[44] L. F. Coffin Jr, R. P. Wesley, N. Y. Schenectady, Apparatus For Study of Effects of Cyclic Thermal Stresses on Ductile Metals, Transactions of the ASME, 1953, Paper no. 53-A-77.

[45] L. F. Coffin Jr, N. Y. Schenectady, A Study of the Effects of Cyclic Thermal Stresses on a Ductile Metal, Transactions of the ASME, 1953, Paper no. 53-A-76.

[46] G. C. Sih, On the Singular Character of Thermal Stresses Near a Crack Tip, Journal of Applied Mechanics, vol. 29, pp. 587-589, 1962.

[47] A. Emery, G. Walker Jr., J. Williams, A Green's Function for the Stress-Intensity Factors of Edge Cracks and its Application to Thermal Stresses, Transactions of the ASME, Journal of Basic Engineering, vol. 91, 1969, pp. 618-624.

[48] K. Kokini, Thermal Shock of a Cracked Strip: Effect of Temperature Dependant Material Properties, Engineering Fracture Mechanics, vol. 25, no. 2, pp. 167-176, 1986.

[49] D. A. Miller, R. H. Priest in High Temperature Fatigue: Properties and Prediction, R. P. Skelton, Ed, Elsevier Science Publishing 1987, pp113-175

[50] D. J. Beauchamp, Thermal-Mechanical Strain cycling at Elevated Temperature, PhD Thesis, University of Bristol

[51] D. J. Marsh, A Thermal Shock Fatigue Study of Type 304 and 316 Stainless Steel, Fatigue of Engineering Materials and Structures, Vol.4, No.2, 1981, pp179-195

[52] D. J. Marsh, F. D. W. Charlesworth, The Determination and Interpretation of Thermally Promoted Crack Initiation and Crack Growth Data and its Correlation with Current Uniaxial Design Data, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown Eds., 1985, pp700-719

[53] D. J. Marsh, Fatigue Crack Initiation and Propagation in Stainless Steels Subjected to Thermal Cycling Conditions, International Conference of Mechanical Behaviour & Nuclear Applications of Stainless Steel at Elevated Temperatures, Varese, Italy, 20-22 May, 1981, pp. 113-121.

[54] Benham & Hoyle, H. G. Baron in Thermal Stress, Pitman & Sons Ltd, 1964, Chp 10 pp 190

[55] L. Northcott, H. G. Baron, The Craze-Cracking of Metals, J. Iron & Steel Inst., Vol.184, 1956, pp385-408

[56] E. Glenny, A Technique for Thermal Shock and Thermal Fatigue Testing Based on the Use of Fluidized Solids, J. Inst. Metals, 1958-59, vol. 87, pp294

[57] S. S. Manson, Thermal Stress and Low Cycle Fatigue, Robert Kreiger Publishing Company, 1981, pp. 275-312.

[58] R. P. Skelton & K. J. Nix, Crack Growth Behaviour in Austenitic Steels During Thermal Quenching from 550°C, High Temperature Technology, vol 5, no. 1 1987, pp. 3-11.

[59] R. I. Ramakrishnan, Quench Analysis of Aerospace Components Using FEM, Proceedings of the 1st International Conference on Quenching & Control of Distortion, Chicago, Illinois, USA, 22-25 September, 1991, pp. 235-242.

[60] B. Askel,, W. R. Arthur, S. Mukherjee, A Study of Quenching: Experiment and Modelling, Journal of Engineering for Industry, vol. 114, August 1992, pp. 309-316.

[61] J. Rasty, G. Roy, D.E. Hunter, Application of Abaqus & Adina Finite Element Codes to the Analysis of Residual Stresses Induced by Quenching, Spring Conference on Experimental Mechanics, Dearborn, Michigan, USA, 7-9 June, 1993, pp. 205-213.

[62] M. Seki, M. Akiba, M. Araki, K. Yokoyama, M. Dairaku, T. Horie, K. Fukaya, M. Ogawa, H. Ise, Thermal Shock Tests on Various Materials of Plasma Facing Components for FER/ITER, (Fusion Experimental Reactor/International Thermonuclear Experimental Reactor), Fusion Engineering & Design, 15, 1991, pp. 59-74.

[63] M. W. Brown, An Analysis of Mode I Fatigue Crack Propagation Under Thermal Downshock, University of Sheffield, Department of Mechanical and Process Engineering.

[64] S. T. Hasan, M. W. Brown, Thermal Downshock Fatigue Testing in AISI 316 Stainless Steel Plate, High Temperature Technology, vol. 8, no. 2, May 1990, pp. 89-97.

[65] S. T. Hasan, M. W. Brown, Fatigue Crack Propagation Due to Thermal Shock in AISI 316 Stainless Steel, 1989, kindly provided by S. T. Hasan, Depertment of Engineering, Sheffield Hallam University.

[66] R. P. Skelton, Introduction to Thermal Shock, High Temperature Technology, Vol.8, No.2, 1990, pp75.

[67] J. Seivers, X. Lui, Fracture Mechanics Analysis Including Constraint Investigations On Reactor Pressure Under Pressurized Thermal Shock Loading, Nuclear Engineering and Design, 158, 1995, pp. 429-435.

[68] D. S. Chawla, S. R. Bhate, H. S. Kushwaha, Numerical Simulation of Crack Growth and Arrest Pressure Vessel Thermal Shock, International Journal of Pressure and Piping, 77, 2000, pp.261-271.

[69] Y. He, T. Isozaki, Fracture Mechanics Analysis and Evaluation for the RPV of the Chinese Qinshan 300 MW NPP under PTS, Nuclear Engineering and Design, 201, 2000, pp. 121-137.

[70] R. Bartsch, M. Wenk, Safety Against Brittle Fracture of the Reactor Pressure Vessel in the Nuclear Power Plant Obrigheim, Nuclear Engineering and Design, 198, 2000, pp. 97-113.

[71] E. Roos, U. Eisele, L. Stumpfrock, Transferability of Results of PTS Experiments to the Integrity Assessment of Reactor Pressure Vessels, Nuclear Engineering and Design, 198, 2000, pp. 173-183.

[72] G. Yagawa, S. Yoshimura, N. Soneda, M. Hirano, Probabilistic Fracture Mechanics Analysis of Nuclear Pressure Vessels Under PTS Events, Nuclear Engineering and Design, 174, 1997, pp. 91-100.

[73] G. Yagawa, Y. Kanto, S. Yoshimura, Probabilistic Fracture Mechanics of Nuclear Structural Components: Consideration of Transition from Embedded Crack to Surface Crack, Nuclear Engineering and Design, 191, 1999, 263-273.

[74] L. G. Zhao, T. J. Lu, N. A. Fleck, Crack Channelling and Spalling in a Plate Due to Thermal Shock Loading, Journal of Mechanics and Physics of Solids, 48, 2000, pp. 867-897.

[75] C. Jang, I. Jeong, S. Hong, The Effect of Flaw Orientation on the Integrity of PWR Pressure Vessel at the Events of Pressurized Thermal Shock, Nuclear Engineering and Design, 197, 2000, pp. 249-264.

[76] M. Matsubara, A Parametric Study on J-Integral Under Pressurized Thermal Shock by Using Statically Indeterminate Fracture Mechanics, Nuclear Engineering and Design, 196, 2000, 153-159.

[77] B. R. Bass, W. J. McAfee, P. T. Williams, W. E. Pennell, Fracture Assessment of Shallow-Flaw Cruciform Beams Under Uniaxial and Biaxial Loading Conditions, Nuclear Engineering and Design, 188, 1999, pp. 259-288.

[78] Benham & Hoyle, E. E. James in Thermal Stress, Pitman & Sons Ltd, 1964, Chp 6 pp 100.

[79] B. A. Boley, The Determination of Temperature, Stresses and Deflections in Two-Dimensional Thermoelastic Problems, Journal of Aeronautical Sciences, vol. 23, no. 1, 1956, pp. 67-75

[80] D. K. Brown, Multivariable Approach to the Finite Difference Solution of Elasto-Plastic Problems, International Journal for Numerical Methods in Engineering, vol. 10, pp. 361-377, 1976

[81] R. W. Clough, The Finite Element Method in Plane Stress Analysis, 2nd Conference on Electronic Computation, Pittsburgh, Pa, Sept, 1960, pp. 345-378.

[82] A. Hrennikoff, Solution of Problems of Elasticity by the Framework Method, Journal of Applied Mechanics, vol. A8, pp. 169-175, 1941.

[83] J. K. Sharples, Determination of Stress Intensity Factors for a Plate with a Single Edge Crack Using Finite Element Methods, NEL Report N. 615, Department of Industry May 1976.

[84] S. N. Atluri, M. Nakagaki, W. H. Chen, Fracture Analysis Under Large Scale Plastic Yielding: A finite deformation, embedded singularity, elastoplastic incremental finite element solution, Flaw Growth and Fracture, ASTM STP 631, 1977, pp. 42-61.

[85] R. L. Roche, Use of the Calculation if Integral J_1, Fracture 1977, vol. 3, ICF4, Waterloo, Canada, June 9-24, 1977.

[86] D. M. Parks, The Virtual Crack Extension Method for Nonlinear Material Behaviour, Computer Methods in Mechanics and Engineering, vol. 12, 1977, pp. 353-364.

[87] H. Yuan, D. Kalkhof, Assessment of Cracks Under Transient Thermal-Mechanical Loading Conditions in Ductile Engineering Materials, Proceedings from the 12 Bienniel Conference of Fracture, 14-18 Sept. 1998, EDF-12 Fracture From Defects, Eds. M. W. Brown, E. R. de los Rios, K. J. Miller. pp. 435-440.

[88] A. S. Kuo, An Experimental and FEM Study on Crack Opening Displacement, Fracture 1977, vol. 3, ICF4, Waterloo, Canada, June 9-24, 1977.

[89] M. Nakagaki, W. H. Chen, S. N. Atluri, A Finite Element Analysis of Stable Crack Growth – I, Elastic-Plastic Fracture, ASTM STP 668, J. D. Landes, J. A. Begley, G. A. Clarke, Eds., 1979, pp. 195-213.

[90] M. G. Dawes, Elastic-Plastic Fracture Toughness Based on COD and J-Contour Integral Concepts, Elastic-Plastic Fracture, ASTM STP 668, J. D. Landes, J. A. Begley, G. A. Clarke, Eds., 1979, pp. 307-333.

[91] C. Berger, H. P. Keller, D. Munz, Determination of Fracture Toughness with Linear-Elastic and Elastic-Plastic Methods, Elastic-Plastic Fracture, ASTM STP 668, J. D. Landes, J. A. Begley, G. A. Clarke, Eds., 1979, pp. 307-333.

[92] B. A. Bilby, G. E. Cardew, M. R. Goldthorpe, I. C. Howard, A Finite Element Investigation of the Effect of Specimen Geometry on the Field of Stress and Strain at the Tips of Stationary Cracks, Department of Mechanical Engineering, University of Sheffield.

[93] S. H. Ju, B. I. Sandor, M. E. Plesha, Life Prediction of Solder Joints by Damage and Fracture Mechanics, EEP- vol. 7, Structural Analysis in Microelectronics and Fibre Optics, New Orleans, USA, 28 Nov. – 3 Dec. 1993, ASME, pp. 105-111.

[94] P. Reimers, Simulation of Mixed Mode Crack Growth, Computers & Structures, Pergamon Press, vol. 40, no. 2, pp. 339-346, 1991.

[95] Ming-Chang Jeng, Ji-Liang. Doong, Wei-Chung Liu, Finite Element Analysis of Crack Growth Life Prediction Under Complex Load History, Engineering Fracture Mechanics, vol. 46, no. 4, 1993, pp. 607-616.

[96] M. Ayari, B. Sun, T. Hsu, A Continuum Damage Mechanics Model for Cyclic Creep Fracture, Engineering Fracture Mechanics, vol. 47, no. 2, 1994, pp. 215-222.

[97] D. R. Hayhurst, High Temperature Design and Life Assessment of Structures Using Continuum Damage Mechanics, Creep & Fatigue Conference: Design and Life Assessment at High Temperature, Mechanical Engineering Publications Ltd, 1996, pp. 339-410.

[98] C. Chow, L. Yu, A Unified Approach to Metal Fatigue Based on the Theory of Damage Mechanics, Applications of Continuum Damage Mechanics to Fatigue and Fracture, Orlando, Florida, 21 May 1997, pp. 165-185.

[99] B. A. Bilby, I. C. Howard, Z. H. Li, M. A. Sheikh, Some Experience in the Use of Damage Mechanics to Simulate Crack Behaviour in Specimens and Structures, International Journal of Pressure Vessels and Piping, 64, 1995, pp. 213-223.

[100] Z. H. Li, I. C. Howard, B. A. Bilby, Three-Dimensional Damage Theory of Crack Growth in Large Centre-Cracked Panels Using the Element Removal Technique, Nuclear Engineering and Design, 172, 1997, pp. 1-12.

[101] W. X. Chu, D. J. Smith, On the Use of Displacement Extrapolation to Obtain Crack Tip Singular Stresses and Stress Intensity Factors, Engineering Fracture Mechanics, Vol. 51, No. 3, pp. 391-400, 1995.

[102] D. M. Tracey, Finite Elements for Determination of Crack Tip Elastic Stress Intensity Factors, Engineering Fracture Mechanics, vol. 3, pp. 255-265, 1971.

[103] R. D. Henshell, K. G. Shaw, Crack Tip Finite Element are Unnecessary, International Journal for Numerical Methods in Engineering, vol. 9, 1975, pp. 495-507.

[104] R. S. Barsoum, On the use of Isoparametric Finite Elements in Linear Fracture Mechanics, International Journal for Numerical Methods in Engineering, vol. 10, pp. 25-37, 1976.

[105] V. E. Saouma, D. Schwemmer, Numerical Evaluation of the Quarter-Point Crack Tip Element, International Journal for Numerical methods in Engineering, vol. 20, pp. 1629-1641, 1984.

[106] H. F. Nied, Thermal Shock Fracture in an Edge-Cracked Plate, Journal of Thermal Stresses, vol. 6, pp. 217-229, 1983.

[107] H. F. Nied, Thermal Shock in an Edge-Cracked Plate Subjected to Uniform Surface Heating, Engineering Fracture Mechanics, vol. 26, no. 2, pp. 239-246, 1987.

[108] A. El-Fattah, A. Rizk, S. F. Radwan, Fracture of a Plate Under Transient Thermal Stresses, Journal of Thermal Stresses, vol. 16, pp. 79-102, 1993

[109] A El-Fattah, A. Rizk, A Cracked Plate Under Transient Thermal Stresses Due to Surface Heating, Engineering Fracture Mechanics, vol. 45, no. 5 pp. 687-696, 1993.

[110] F. Xuejun, Y. Shouwen, Thermal Shock Fracture in a Surface Cracked Plate, Engineering Fracture Mechanics, vol. 41, no. 2 pp. 223-228, 1992.

[111] Chwan-Huei Tsai, Chien-Ching Ma, Thermal Weighting Function of Cracked Bodies Subjected to Thermal Loading, Engineering Fracture Mechanics, vol. 41, no. 1, 1992, pp. 27-40.

[112] Kang Yong Lee, Kwan-Bo Sim, Thermal Shock Stress Intensity Factor by Bueckner's Weight Function Method, Engineering Fracture Mechanics, vol. 37, no. 4, pp. 799-804, 1990.

[113] S. Shimamura, Y. Sotoike, Computer Modelling of Thermal Shock-Induced Crack Growth in Brittle materials, Journal of Materials Research, vol. 7, no. 5, May 1995, pp. 1286-1291.

[114] A. G. Tomba, A. L. Cavalieri, Alumina Disks with Different Surface Finish: Thermal Shock Behaviour, Journal of the European Ceramic Society, 20, 2000, pp. 889-893.

[115] A. G. Tomba, A. L. Cavalieri, Numerical Simulation of the Thermal Shock of Alumina Disks with Different Surface Finish, Journal of the European Ceramic Society, 21, 2001, pp. 1205-1212.

[116] Klod Kokini, On The Use of The Finite Element Method For The Solution of a Cracked Strip Under Thermal Shock, Engineering Fracture Mechanics, vol. 24, no. 6, pp. 843-850, 1986

[117] W. H. Chen, K. Ting, Finite Element Analysis of Mixed-Mode Thermoelastic Fracture Problems, Nuclear Energy & Design, vol. 90, pp. 55-65, 1985.

[118] D. E. Katsareas, N. K. Anifantis, On the Computation of Mode I and II Thermal Shock Stress Intensity Factors Using a Boundary-Only Element Method, International Journal for Numerical Methods in Engineering, vol. 38, 1995, pp. 4157-4169.

[119] A. F. Emery, P. K. Neighbors, A. S. Kobayashi, W. J. Love, Stress Intensity Factors in Edge-Cracked Plates Subjected to Transient Thermal Singularities, Trans. ASME, Journal of Pressure Vessel Technology, February 1977, pp. 100-105.

[120] A. F. Emery, The Use of Singularity Programming in Finite-Difference and Finite-Element Computations of Temperature, Journal of Heat Transfer, Transactions of the ASME, August 1973, pp. 344-351.

[121] Gao Hua, Nejat Alagok, Michael Brown and Keith J. Miller, Growth of Fatigue Cracks Under Combined Mode I and Mode II Loads, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown, Eds. American Society for Testing and Materials, Philadelphia, 1985, pp. 184-202.

[122] Leslie P. Pook, Comments on Fatigue Crack Growth Under Mixed Modes I and III and Pure III Loading, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W.

Brown, Eds. American Society for Testing and Materials, Philadelphia, 1985, pp. 249-263.

[123] Gao Hua, Upul S. Fernando, Effect of Non-Proportional Overloading on Fatigue Life, Fatigue and Fracture in Engineering Materials and Structures, vol. 19, pp. 1197-1206, 1996

[124] S. T. Hasan, Analysis of fatigue crack propagation due to cyclic thermal and mechanical stresses, Thesis, Sheffield University, 1990

[125] ASTM Standard, E647, vol. 03.01, 1998

[126] D. Rhodes, J. C. Radon, Effect of Local stress Biaxiality on the Behaviour of Fatigue Crack Growth Test Specimen, Multiaxial Fatigue, ASTM STP 853, K. J. Miller, M. W. Brown, Eds. American Society for Testing and Materials, Philadelphia, 1985, pp. 153-163.

[127] Eastop and McConkey, Applied Thermodynamics for Engineering Technologies, 5th Edition, Longman Scientific & Technical, 1993, pp. 581-2

[128] S. P. Timoshenko, Theory of Elasticity, 3rd Ed., McGraw-Hill

[129] ABAQUS v5.7 User Manuals

T_{MAX} (°C)	T_{MIN} (°C)	ΔT (°C)	t_P (s)	Crack Type	Crack Length *a* (mm)	Mechanical Load (MPa)
625	225	400	3	SEN	1	0 & 120
-	-	-	-	-	2	0 & 120
-	-	-	-	-	3	0 & 120
-	-	-	-	-	4	0 & 120
-	-	-	-	-	5	0 & 120
-	-	-	-	-	6	0 & 120
-	-	-	-	-	7	0 & 120
650	350	300	5	SEN	1	-
-	-	-	-	-	2	-
-	-	-	-	-	3	-
-	-	-	-	-	4	-
-	-	-	-	-	5	-
-	-	-	-	-	6	-
-	-	-	-	-	7	-
550	350	300	5	SEN	1	-
-	-	-	-	-	2	-
-	-	-	-	-	3	-
-	-	-	-	-	4	-
-	-	-	-	-	5	-
-	-	-	-	-	6	-
-	-	-	-	-	7	-
612	300	300	3	DEN	1	0 & 75
-	-	-	-	-	2	0 & 75
-	-	-	-	-	3	0 & 75
-	-	-	-	-	4	0 & 75
-	-	-	-	-	5	0 & 75
-	-	-	-	-	6	0 & 75
-	-	-	-	-	7	0 & 75

Table 1: Types of thermal shock loading conducted.

Biaxial State	Stress Range $\Delta\sigma$ (MPa)	R-Ratio	T (°C)	Crack Type	Crack Length a (mm)
Unibiaxial	118.8	0.1	650	CCP	1
-	118.8	0.1	650	CCP	2
-	118.8	0.1	650	CCP	3
-	118.8	0.1	650	CCP	4
-	118.8	0.1	650	CCP	5
-	118.8	0.1	650	CCP	8
-	118.8	0.1	650	CCP	11
Unibiaxial	200.0	0.1	650	CCP	1
-	200.0	0.1	650	CCP	2
-	200.0	0.1	650	CCP	3
-	200.0	0.1	650	CCP	4
-	200.0	0.1	650	CCP	5
-	200.0	0.1	650	CCP	8
-	200.0	0.1	650	CCP	11
Equibiaxial	118.8	0.1	650	CCP	1
-	118.8	0.1	650	CCP	2
-	118.8	0.1	650	CCP	3
-	118.8	0.1	650	CCP	4
-	118.8	0.1	650	CCP	5
-	118.8	0.1	650	CCP	8
-	118.8	0.1	650	CCP	11
Equibiaxial	200.0	0.1	650	CCP	1
-	200.0	0.1	650	CCP	2
-	200.0	0.1	650	CCP	3
-	200.0	0.1	650	CCP	4
-	200.0	0.1	650	CCP	5
-	200.0	0.1	650	CCP	8
-	200.0	0.1	650	CCP	11

Table 2: Loading of cruciform models conducted.

Distance	Cycle Time (s)													
(mm)	0.5	1	2	3	4	5	6	8	10	15	20	30	50	60
0	591	520	426	363	335	310	332	433	485	542	568	600	626	627
1	601	543	463	409	377	352	358	438	485	541	568	599	625	628
3	619	592	538	498	461	433	413	455	492	544	571	601	626	631
5	629	617	584	552	524	501	474	489	514	558	582	613	632	637
7	627	619	602	574	558	539	530	521	535	568	589	612	630	634
10	637	632	624	614	602	590	574	566	570	590	606	626	641	644
15	645	641	636	631	625	618	607	604	604	610	618	631	647	650
20	652	648	643	638	634	629	621	621	622	624	629	640	654	657
25	653	649	644	640	635	632	624	625	626	630	635	644	655	657
30	648	645	640	636	631	625	617	617	620	627	631	638	650	653
35	653	649	644	639	632	626	618	620	623	630	634	640	655	658
39	643	636	627	620	613	608	603	611	616	624	630	639	648	649

Table 3: Experimental space-time thermal distribution by Hasan, [124].

Temperature (°C)	Cyclic Modulus (Gpa)	Strength Coefficient (MPa)	Strain Hardening Exponent	0.2% Proof Stress
23	210.0	2000	3.356	313.8
400	171.5	1997	3.339	310.4
500	178.0	1281	5.013	370.6
600	168.5	1246	4.435	306.8
700	175.5	684	5.025	198.6

Table 4: Cyclic material properties for AISI 316 – Ramberg-Osgood.

Temperature (°C)	Cyclic Modulus (Gpa)	Yield Offset, y_o, (MPa)	0.2% Proof Stress	Strain Hardening Exponent
23	210.0	1.338	313.8	3.356
400	171.9	1.106	310.4	3.340
500	180.2	0.957	370.6	5.033
600	166.8	1.036	306.8	4.505
700	168.0	1.630	198.6	5.067

Table 5: Cyclic material properties for AISI 316 – Deformation Plasticity.

SEN 625-225°C in 3 seconds – 0MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
1.08	0	0.000	0.00E+00
1.14	595	0.000	0.00E+00
1.48	1449	0.000	0.00E+00
1.53	1732	1.553	2.86E-04
1.89	2860	1.887	2.71E-04
1.98	3206	1.940	2.70E-04
2.24	4476	2.291	2.85E-04
2.32	4712	2.330	2.84E-04
2.68	5674	2.623	2.58E-04
2.73	5965	2.730	2.74E-04
2.79	6215	2.793	2.58E-04
2.94	7122	2.977	2.06E-04
3.10	7448	3.048	2.05E-04
3.23	8583	3.270	2.02E-04
3.38	8994	3.360	1.95E-04
3.54	10054	3.550	1.87E-04
3.67	10466	3.631	1.98E-04
3.80	11514	3.845	1.81E-04
3.92	11934	3.929	1.85E-04
4.19	13143	4.134	1.62E-04
4.33	14570	4.362	1.43E-04
4.51	15887	4.527	1.27E-04
4.59	16362	4.570	1.33E-04
4.69	17299	4.702	1.44E-04
4.76	17759	4.774	1.38E-04
4.92	18793	4.916	1.33E-04
5.03	19404	5.006	1.28E-04
5.09	20266	5.098	1.20E-04
5.15	20679	5.148	1.11E-04

5.27	22052	5.300	1.01E-04
5.51	23876	5.461	9.85E-05
5.59	25297	5.605	9.06E-05
5.67	26590	5.723	8.36E-05
5.89	27960	5.828	7.34E-05
5.93	29472	5.940	6.89E-05
6.02	30902	6.021	6.20E-05
6.10	32448	6.103	4.90E-05
6.22	35274	6.241	4.84E-05
6.35	36890	6.317	4.50E-05
6.41	39766	6.440	4.23E-05
6.52	41264	6.502	3.95E-05
6.54	42578	6.548	3.48E-05
6.63	44362	6.621	3.24E-05
6.66	45764	6.648	2.89E-05
6.68	47004	6.684	2.93E-05
6.73	49949	6.771	2.66E-05
6.84	51326	6.807	2.82E-05
6.86	52780	6.851	2.95E-05
6.99	58717	7.021	3.15E-05
7.08	60194	7.056	3.87E-05
7.10	61582	0.000	0.00E+00
7.18	63090	0.000	0.00E+00
7.32	65705	0.000	0.00E+00

Table 6: Crack gorwth data for single edge thermal shock test 625-225°C in 3 seconds with no uniaxial dead load [124].

SEN 625-225□C in 3 seconds – 0MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
1.05	0	0.000	0.00E+00
1.10	375	0.000	0.00E+00
1.36	1323	0.000	0.00E+00
1.75	1779	1.637	3.73E-04
1.96	2772	2.027	4.02E-04
2.25	3221	2.236	4.12E-04
2.80	4694	2.796	3.80E-04
3.45	6185	3.399	3.65E-04
3.66	7122	3.721	3.46E-04
3.92	7636	3.902	3.37E-04
4.20	8560	4.186	3.28E-04
4.37	9088	4.384	3.25E-04
4.65	10027	4.666	2.94E-04
4.85	10504	4.809	2.89E-04
5.05	11440	5.065	2.79E-04
5.18	11898	5.156	2.89E-04
5.44	12928	5.542	2.86E-04
5.62	13393	5.681	3.04E-04
6.26	14896	6.069	2.47E-04
6.43	17245	6.507	1.87E-04
6.71	18712	6.725	1.81E-04
6.90	20147	6.931	1.75E-04
7.23	21578	0.000	0.00E+00
7.56	23043	0.000	0.00E+00
7.87	24800	0.000	0.00E+00

Table 7: Crack growth data for single edge thermal shock test 625-225°C in 3 seconds with 120MPa uniaxial dead load [124].

DEN 612-300°C in 5s – 0MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
1.12	0	0.000	0.00E+00
1.26	487	0.000	0.00E+00
1.55	1119	0.000	0.00E+00
1.68	1327	1.706	5.19E-04
1.82	1512	1.820	5.54E-04
2.06	1802	2.003	6.00E-04
2.36	2550	2.406	4.57E-04
2.47	2738	2.493	3.82E-04
2.64	3005	2.580	3.24E-04
2.67	3257	2.627	2.98E-04
2.76	3994	2.807	4.63E-04
2.85	4245	2.954	3.37E-04
3.10	4491	3.052	3.33E-04
3.32	4731	3.215	4.48E-04
3.33	5406	3.396	1.50E-04
3.35	5598	3.365	2.18E-04
3.43	5851	3.406	1.98E-04
3.44	6092	3.494	2.75E-04
3.71	6785	3.700	2.64E-04
3.82	7145	3.787	2.56E-04
3.87	7567	3.892	2.47E-04
3.99	8217	4.027	2.01E-04
4.12	8585	4.088	2.25E-04
4.20	8946	4.174	2.48E-04
4.27	9641	4.377	2.26E-04
4.49	9959	4.425	2.09E-04
4.57	10335	4.503	2.03E-04
4.70	11880	4.795	1.64E-04
4.96	12691	4.897	1.38E-04

5.09	13956	5.066	1.34E-04
5.12	14568	5.166	1.16E-04
5.23	15305	5.228	9.14E-05
5.32	15853	5.280	9.02E-05
5.34	16706	5.354	8.00E-05
5.39	17497	5.400	5.89E-05
5.51	19565	5.484	7.03E-05
5.52	20243	5.528	9.17E-05
5.54	20929	5.597	9.25E-05
5.71	21750	5.716	1.03E-04
5.88	22970	5.833	9.58E-05
5.91	23869	5.876	8.07E-05
6.00	26627	0.000	0.00E+00
6.09	28046	0.000	0.00E+00
6.28	29423	0.000	0.00E+00

Table 8: Crack growth data for double edge symmetrical thermal shock test 612-300°C in 5 seconds with zero uniaxial dead load [124].

DEN 612-300°C in 5s – 120MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
1.47	0	0.000	0.00E+00
1.60	126	0.000	0.00E+00
1.81	290	0.000	0.00E+00
2.08	417	2.226	2.34E-03
3.18	708	2.901	2.82E-03
4.96	1355	5.210	3.26E-03
5.54	1495	5.812	2.65E-03
6.73	1673	6.277	2.48E-03
7.27	2031	7.170	2.18E-03
7.65	2776	8.109	1.42E-03
8.06	2903	8.141	2.54E-03

8.67	3068	8.517	3.01E-03
8.92	3206	9.165	3.33E-03
10.01	3442	9.912	2.99E-03
10.38	3577	10.239	2.68E-03
11.57	4234	11.557	1.59E-03
11.97	4540	11.706	1.01E-03
12.19	5613	12.335	3.41E-04
12.22	6106	12.274	1.84E-04
12.32	6961	12.365	1.55E-04
12.41	7174	12.392	1.33E-04
12.44	7384	12.428	1.34E-04
12.48	7652	12.461	1.40E-04
12.53	8394	12.525	1.44E-04
12.55	8715	12.569	1.82E-04
12.64	9036	12.675	1.83E-04
12.84	9793	12.859	2.02E-04
12.98	10126	12.909	1.92E-04
13.04	11182	13.051	1.24E-04
13.07	11873	13.108	8.34E-05
13.19	12590	13.151	5.91E-05
13.23	14194	13.235	4.16E-05
13.24	14547	0.000	0.00E+00
13.28	16812	0.000	0.00E+00
13.30	17400	0.000	0.00E+00

Table 9: Crack growth data for double edge symmetrical thermal shock test 612-300°C in 5 seconds with 0MPa uniaxial dead load [124].

Unibiaxial - $\Delta\sigma$ = 118.8MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
2.30	4560	0.00	0.00E+00
2.63	8160	0.00	0.00E+00
3.10	11760	0.00	0.00E+00
3.52	14580	3.51	1.56E-04
3.56	14814	3.56	1.52E-04
3.72	15894	3.72	1.58E-04
3.89	16974	3.89	1.61E-04
4.07	18054	4.07	1.69E-04
4.26	19134	4.26	1.76E-04
4.44	20214	4.45	1.83E-04
4.64	21186	4.63	1.93E-04
4.65	21294	4.65	1.92E-04
4.86	22374	4.88	1.96E-04
5.10	23454	5.10	1.83E-04
5.31	24534	5.29	1.65E-04
5.42	25614	5.47	1.92E-04
5.44	25614	5.46	1.90E-04
5.51	25884	5.51	1.86E-04
5.76	26964	5.74	2.34E-04
6.00	28044	6.00	2.50E-04
6.27	29124	6.28	2.67E-04
6.60	30204	6.59	2.82E-04
6.91	31284	6.91	2.99E-04
7.24	32364	7.25	2.72E-04
7.56	33444	7.54	2.25E-04
7.64	33642	7.60	2.16E-04
7.73	34524	7.78	2.66E-04
7.73	34524	7.77	2.24E-04
7.84	34830	7.85	2.55E-04

8.22	35910	8.21	3.49E-04
8.58	36990	8.59	3.63E-04
9.00	38070	9.01	3.58E-04
9.40	39150	0.00	0.00E+00
9.84	40230	0.00	0.00E+00
10.09	41310	0.00	0.00E+00

Table 10: Crack growth data for isothermal unibiaxial cruciform test at $\Delta\sigma = 118.8$MPa [124].

Unibiaxial - $\Delta\sigma = 200.0$MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
3.67	409	0.00	0.00E+00
3.69	634	0.00	0.00E+00
3.72	694	0.00	0.00E+00
3.76	814	3.75	2.90E-04
3.79	934	3.79	3.85E-04
3.84	1054	3.84	4.36E-04
3.92	1276	3.95	6.38E-04
3.99	1294	3.96	6.21E-04
4.04	1414	4.04	7.46E-04
4.13	1511	4.12	8.08E-04
4.12	1534	4.13	8.08E-04
4.25	1654	4.24	9.79E-04
4.27	1696	4.30	9.81E-04
4.45	1834	4.43	9.68E-04
4.45	1852	4.37	6.45E-04
4.73	2194	4.87	1.08E-03
4.82	2314	4.97	9.52E-04
5.37	2434	5.02	7.35E-04
5.45	3842	5.51	2.55E-04
5.55	4202	5.57	2.36E-04

5.64	4562	5.63	2.50E-04
5.75	4922	5.73	2.90E-04
5.74	4964	5.74	2.80E-04
5.81	5282	5.84	2.98E-04
5.97	5642	5.95	3.18E-04
6.07	6002	6.07	3.56E-04
6.18	6362	6.20	3.84E-04
6.27	6542	6.26	3.81E-04
6.50	7082	6.50	4.29E-04
6.66	7442	6.66	4.49E-04
6.82	7802	6.82	4.51E-04
6.98	8162	6.98	4.76E-04
7.16	8522	7.15	4.99E-04
7.31	8882	7.34	5.13E-04
7.55	9242	7.53	5.38E-04
7.74	9602	7.72	5.73E-04
7.91	9962	7.93	6.07E-04
8.15	10322	8.15	6.20E-04
8.41	10682	8.38	6.74E-04
8.64	11042	8.64	6.92E-04
8.67	11144	8.71	6.87E-04
8.91	11402	8.88	6.88E-04
9.14	11762	9.15	7.07E-04
9.39	12122	9.39	7.31E-04
9.62	12446	9.62	8.12E-04
9.64	12482	0.00	0.00E+00
9.95	12842	0.00	0.00E+00
10.33	13202	0.00	0.00E+00

Table 11: Crack growth data for isothermal unibiaxial cruciform test at $\Delta\sigma = 200.0$MPa [124].

Equibiaxial - $\Delta\sigma$ = 118.8MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
2.18	271792	0.00	0.00E+00
2.20	272332	0.00	0.00E+00
2.29	275392	0.00	0.00E+00
2.40	278992	2.41	4.48E-05
2.57	282412	2.58	6.24E-05
2.65	283546	2.65	7.14E-05
2.73	284626	2.73	7.29E-05
2.83	285706	2.82	7.62E-05
2.91	286786	2.90	7.76E-05
2.98	287866	2.99	8.08E-05
3.07	288946	3.07	8.20E-05
3.16	290026	3.16	8.65E-05
3.27	291106	3.26	9.14E-05
3.35	292186	3.36	9.21E-05
3.47	293266	3.46	9.26E-05
3.57	294346	3.56	9.29E-05
3.66	295426	3.66	9.60E-05
3.77	296506	3.76	9.78E-05
3.87	297586	3.87	1.01E-04
3.99	298666	3.99	1.05E-04
4.10	299746	4.10	1.09E-04
4.21	300826	4.22	1.15E-04
4.34	301906	4.34	1.16E-04
4.47	302986	4.47	1.20E-04
4.62	304066	4.61	1.24E-04
4.73	305146	4.74	1.27E-04
4.88	306226	4.88	1.30E-04
5.02	307306	5.02	1.33E-04
5.17	308386	5.17	1.39E-04

5.32	309466	5.32	1.43E-04
5.47	310546	5.47	1.50E-04
5.64	311626	5.64	1.56E-04
5.81	312706	5.81	1.62E-04
6.01	313786	5.99	1.69E-04
6.17	314866	6.18	1.69E-04
6.35	315946	6.36	1.74E-04
6.57	317026	6.55	1.79E-04
6.72	318106	6.75	1.86E-04
6.96	319186	6.95	1.91E-04
7.16	320266	7.16	2.12E-04
7.23	320608	7.23	2.00E-04
7.26	320806	7.27	2.00E-04
7.40	321346	7.38	2.05E-04
7.48	321886	7.49	2.05E-04
7.60	322426	7.60	2.06E-04
7.82	323506	7.82	2.08E-04
8.06	324586	8.05	2.20E-04
8.29	325666	8.30	2.26E-04
8.54	326746	8.54	2.36E-04
8.82	327826	8.80	2.42E-04
8.96	328510	8.97	2.49E-04
9.08	328906	9.07	2.50E-04
9.34	329986	9.34	2.57E-04
9.64	331066	0.00	0.00E+00
9.91	332146	0.00	0.00E+00
10.22	333226	0.00	0.00E+00

Table 12: Crack growth data for isothermal equibiaxial cruciform test at $\Delta\sigma$ = 118.8MPa [124].

Equibiaxial - $\Delta\sigma$ = 200.0MPa			
Experimental		ASTM E647	
a (mm)	N	a' (mm)	da/dN
3.43	561	0.00	0.00E+00
3.46	926	0.00	0.00E+00
3.52	1329	0.00	0.00E+00
3.57	1466	3.57	2.95E-04
3.61	1586	3.61	3.03E-04
3.67	1706	3.65	3.28E-04
3.68	1826	3.69	3.44E-04
3.70	1899	3.72	3.05E-04
3.74	1946	3.73	3.04E-04
3.78	2066	3.77	3.44E-04
3.79	2186	3.81	3.25E-04
3.86	2306	3.84	2.74E-04
4.05	3101	4.04	2.62E-04
4.04	3146	4.06	2.60E-04
4.08	3206	4.07	3.88E-04
4.10	3266	4.09	4.12E-04
4.11	3326	4.12	4.43E-04
4.16	3386	4.15	3.55E-04
4.18	3446	4.17	3.81E-04
4.19	3506	4.19	3.84E-04
4.19	3566	4.21	3.40E-04
4.24	3621	4.24	3.84E-04
4.28	3746	4.28	4.13E-04
4.35	3866	4.34	4.25E-04
4.36	3986	4.38	3.75E-04
4.44	4106	4.42	4.33E-04
4.48	4226	4.47	4.27E-04
4.49	4300	4.51	4.35E-04
4.59	4420	4.56	4.05E-04

4.54	4432	4.56	4.25E-04
4.65	4631	4.65	4.91E-04
4.68	4720	4.68	5.96E-04
4.78	4840	4.76	4.65E-04
4.76	4854	4.77	5.39E-04
4.82	4900	4.79	5.52E-04
4.82	5020	4.86	4.64E-04
4.93	5140	4.90	4.68E-04
4.97	5260	4.96	4.81E-04
4.99	5380	5.02	5.12E-04
5.10	5500	5.08	5.27E-04
5.13	5584	5.13	5.64E-04
5.15	5620	5.15	6.02E-04
5.21	5740	5.21	5.52E-04
5.29	5860	5.29	5.59E-04
5.30	5886	5.30	5.25E-04
5.36	5980	5.35	5.83E-04
5.39	6049	5.39	5.60E-04
5.39	6100	5.42	5.73E-04
5.51	6220	5.49	5.59E-04
5.54	6304	5.54	5.41E-04
5.54	6340	5.55	5.91E-04
5.61	6460	5.61	5.48E-04
5.67	6580	5.69	6.30E-04
5.81	6700	5.77	6.81E-04
5.84	6820	5.86	7.29E-04
5.93	6940	5.95	6.22E-04
6.00	7003	5.98	5.60E-04
6.01	7011	5.98	5.54E-04
6.12	7305	6.13	4.87E-04
6.52	8091	6.52	4.82E-04
6.70	8451	6.69	5.15E-04
6.88	8811	6.89	5.12E-04

7.03	9171	7.07	5.76E-04
7.13	9186	7.08	5.60E-04
7.25	9531	7.28	6.13E-04
7.54	9891	7.51	6.57E-04
7.74	10251	7.74	7.35E-04
8.05	10607	8.03	7.93E-04
7.98	10611	8.04	7.57E-04
8.35	10971	8.33	8.09E-04
8.64	11331	8.60	8.13E-04
8.87	11691	8.89	8.75E-04
9.16	12051	9.20	8.92E-04
9.55	12411	0.00	0.00E+00
9.95	12771	0.00	0.00E+00
10.25	13131	0.00	0.00E+00

Table 13: Crack growth data for isothermal equibiaxial cruciform test at $\Delta\sigma = 200.0$MPa [124].

Number of Contours	*Mesh Density*	*J - Integral (N/mm)*	*K_I ($MPa.m^{3/2}$)*		*Crack Tip Stress (MPa)*		*Plastic Zone Size (mm)*	
			By J	*By U_y*	σ_x	σ_y	r_{px}	r_{py}
8	10.7	0.5508	9.5	9.5	3380	3524	0.172	0.350
12	19.0	0.5508	9.5	9.6	5126	5283	0.172	0.355
20	43.3	0.5508	9.5	9.6	8653	8794	0.180	0.355
30	130	0.5508	9.5	9.6	13074	13175	0.180	0.355

Table 14: Linear-elastic finite elements stress intensity and plastic zone sizes for thin centre-cracked square plate model verification.

K_I/K_0	K_I ($MPa.m^{3/2}$)	*Plastic Zone Size (mm)*	
		r_{px}	r_{py}
1.005	9.4	0.189	0.347

Table 15: Classical solution by Rooke & Cartwright for CCP.

Number of Contours	*Mesh Density*	*J - Integral (N/mm)*	*K_I ($MPa.m^{-3/2}$)*		*Crack Tip Stress (MPa)*		*Plastic Zone Size (mm)*	
			By J	*By U_y*	σ_x	σ_y	r_{px}	r_{py}
8	10.7	0.8025	11.5	11.6	4129	4256	0.355	0.537
12	19.0	0.8026	11.5	11.5	6237	6378	0.359	0.525
20	43.3	0.8026	11.5	11.6	10495	10616	0.346	0.511
30	130	0.8026	11.5	11.3	15832	15904	0.339	0.516

Table 16: Linear-elastic finite elements stress intensity and plastic zone sizes for thin edge-cracked rectangular plate model verification – pin-joint constraint.

Number of Contours	*Mesh Density*	*J - Integral (N/mm)*	K_I *(MPa.m$^{-3/2}$)*		*Crack Tip Stress (MPa)*		*Plastic Zone Size (mm)*	
			By J	*By U_y*	σ_x	σ_y	r_{px}	r_{py}
8	10.7	0.8025	11.5	11.6	4129	4256	0.357	0.537
12	19.0	0.8026	11.5	11.5	6237	6378	0.349	0.501
20	43.3	0.8026	11.5	11.6	10495	10616	0.345	0.520
30	130	0.8026	11.5	11.3	15832	15904	0.349	0.516

Table 17: Linear-elastic finite elements stress intensity and plastic zone sizes for thin edge-cracked rectangular plate model verification – built-in constraint.

K_I/K_0	K_I *(MPa.m$^{-3/2}$)*	*Plastic Zone Size (mm)*	
		r_{px}	r_{py}
1.222	11.5	0.256	0.469

Table 18: Classical solution by Rooke & Cartwright for ECP.

# Elements	t' (s)	h (W/m^2K)	t_{max} (s)
5	1.5	5925	1.721352
5	1	6489	1.617898
5	0.1	7781	1.421989
5	0.01	7923	1.403620
5	0.001	7937	1.401511
10	0.5	7057	0.591021
10	0.1	7319	0.579167
10	0.01	7387	0.576175
10	0.001	7395	0.575867
20	0.25	7381	0.200264
20	0.1	7454	0.199501
20	0.01	7519	0.198820
20	0.001	7525	0.198751
50	0.05	7539	0.041637
50	0.001	7557	0.041617
50	0.0001	7558	0.041617
100	0.01	7555	0.011608
100	0.001	7561	0.011606
100	0.0001	7562	0.011606
200	0.0025	7561	0.003080
200	0.001	7562	0.003080
200	0.0001	7563	0.003080

Table 19: Finite difference refinement series for 625-225°C in 3s.

# Elements	t' (s)	h (W/m^2K)	t_{max} (s)
5	2.5	2697	2.691858
5	1	2919	2.589545
5	0.1	3147	2.492460
5	0.01	3171	2.482719
10	0.5	2914	0.863063
10	0.1	2973	0.857346
10	0.01	2987	0.856021
10	0.001	2988	0.855887
20	0.25	2945	0.258185
20	0.1	2966	0.257812
20	0.01	2979	0.257586
20	0.001	2980	0.257563
20	0.0001	2980	0.257561
50	0.05	2971	0.049270
50	0.001	2978	0.046917
50	0.0001	2978	0.046917
100	0.01	2977	0.012297
100	0.001	2978	0.012296
100	0.0001	2978	0.012296
200	0.001	2978	0.003150
200	0.0001	2978	0.003150

Table 20: Finite difference refinement series for 650-350°C in 5s.

# Elements	t' (s)	h (W/m^2K)	t_{max} (s)
5	2.5	1856	3.297714
5	1	2064	3.155243
5	0.1	2208	3.064431
5	0.01	2222	3.055494
10	1	1926	1.021637
10	0.1	2019	1.009054
10	0.01	2028	1.007917
20	0.25	1989	0.291840
20	0.1	2001	0.291556
20	0.01	2016	0.293148
20	0.001	2010	0.291366
20	0.0001	2010	0.291365
50	0.05	2001	0.051366
50	0.01	2005	0.051360
50	0.001	2005	0.051359
50	0.0001	2005	0.051359
100	0.01	2004	0.013289
100	0.001	2005	0.013289
100	0.0001	2005	0.013289
200	0.001	2005	0.003381
200	0.0001	2005	0.003381

Table 21: Finite difference refinement series for 550-350°C in 5s.

625-225°C 3s			650-350°C 5s			550-350°C 5s		
t'	$T_{t'}$	dT/dt'	t'	$T_{t'}$	dT/dt'	t'	$T_{t'}$	dT/dt'
0.1	484	1412	0.1	585	652	0.1	511	388
0.01	574	5137	0.01	628	2185	0.01	537	1278
0.001	608	17031	0.001	643	7042	0.001	546	4096
0.0001	619	58758	0.0001	648	24065	0.0001	549	13969

Table 22: Forward difference determination of initial cooling rate from theoretical solution.

# Elements	t' (s)	h (W/m^2K)	t_{min} (s)
10	0.1	7100	0.587
10	0.5	7400	0.587
10	1.0	7775	0.587
20	0.01	7000	0.147
20	0.1	7100	0.147
20	0.5	7400	0.147
20	1.0	7775	0.147
40	0.025	7000	0.037
40	0.1	7100	0.037
40	1.0	7800	0.037
75	0.005	7000	0.01
75	0.01	7000	0.01
75	0.1	7100	0.01

Table 23: Finite element refinement series for 625-225 in 3s.

Shock Level	625-225 3s		650-350 5s		550-350 5s	
Material Model	Strain Energy (σ-ε)	Spanned Area (σ-x)	Strain Energy (σ-ε)	Spanned Area (σ-x)	Strain Energy (σ-ε)	Spanned Area (σ-x)
Cyclic Elasticity	4.627E+6	5.697E+6	2.662E+6	4.596E+6	1.16E+6	3.055E+6
Neuber's Elastic-Plastic	1.910E+6	2.228E+6	1.200E+6	1.766E+6	783E+3	1.945E+6
Cyclic Ratio	2.423	2.557	2.218	2.603	1.479	1.571
Mechanical Elasticity	2.119E+6	2.609E+6	1.146E+6	1.979E+6	569E+3	1.500E+6
Linear-Elastic Elastic-Plastic	1.728E+6	2.144E+6	1.033E+6	1.770E+6	512E+3	1.500E+6
Mech. Ratio	1.226	1.217	1.109	1.118	1.111	1.000

Table 24: Strain energy and spanned areas for elastic and elastic-plastic material models.

$T_{INITIAL}$ (°C)	T_{END} (°C)	T_A (°C)	ΔT (°C)	T_{FAC}	t_P (s)	h (W/m^2K)	σ_{MAX} (MPa)
750	120	20	630	0.137	3	34946	1445
750	220	20	530	0.274	3	14314	1134
750	320	20	430	0.411	3	7555	897
750	420	20	330	0.548	3	4266	687
750	520	20	230	0.685	3	2330	477
650	120	20	530	0.159	3	27258	1198
650	220	20	430	0.317	3	10874	914
650	320	20	330	0.476	3	5470	693
650	420	20	230	0.635	3	2823	483
650	520	20	130	0.794	3	1262	273
550	120	20	430	0.189	3	20448	957
550	220	20	330	0.377	3	7855	701
550	320	20	230	0.566	3	3628	488
550	420	20	130	0.755	3	1530	276
750	120	20	630	0.137	4	30361	1398
750	220	20	530	0.274	4	12421	1082
750	**320**	**20**	**430**	**0.411**	**4**	**6559**	**845**
750	420	20	330	0.548	4	3684	640
750	520	20	230	0.685	4	2019	445
650	120	20	530	0.159	4	23664	1157
650	220	20	430	0.317	4	9437	869
650	320	20	330	0.476	4	4762	649
650	420	20	230	0.635	4	2448	451
650	520	20	130	0.794	4	1093	255
550	**120**	**20**	**430**	**0.189**	**4**	**17772**	**921**
550	220	20	330	0.377	4	6810	662
550	320	20	230	0.566	4	3144	457
550	420	20	130	0.755	4	1335	260
750	120	20	630	0.137	5	27226	1363

750	220	20	530	0.274	5	11130	1042
750	320	20	430	0.411	5	5852	804
750	420	20	330	0.548	5	3300	603
750	520	20	230	0.685	5	1807	418
650	120	20	530	0.159	5	21179	1124
650	220	20	430	0.317	5	8458	833
650	320	20	330	0.476	5	4262	615
650	420	20	230	0.635	5	2191	425
650	520	20	130	0.794	5	987	241
550	120	20	430	0.189	5	15922	892
550	220	20	330	0.377	5	6100	632
550	320	20	230	0.566	5	2824	431
550	420	20	130	0.755	5	1186	244

Table 25: Finite difference thermal model series.

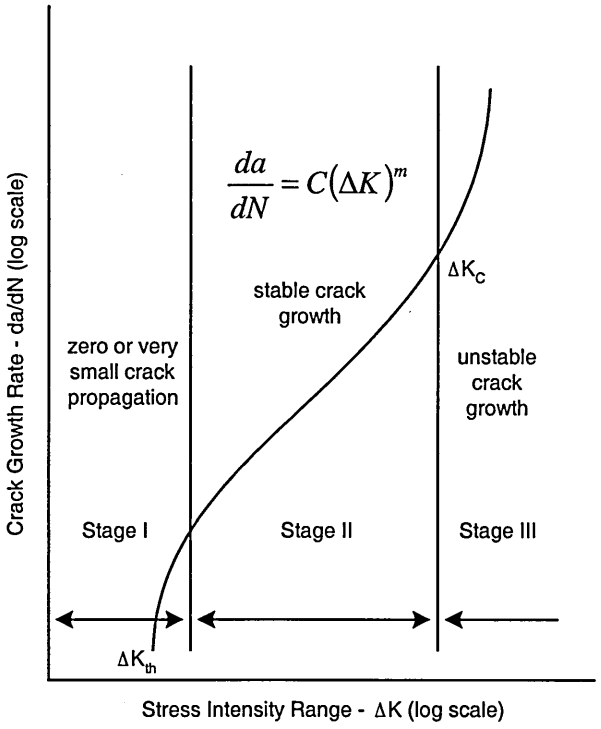


Figure 1: Crack growth rate pattern.

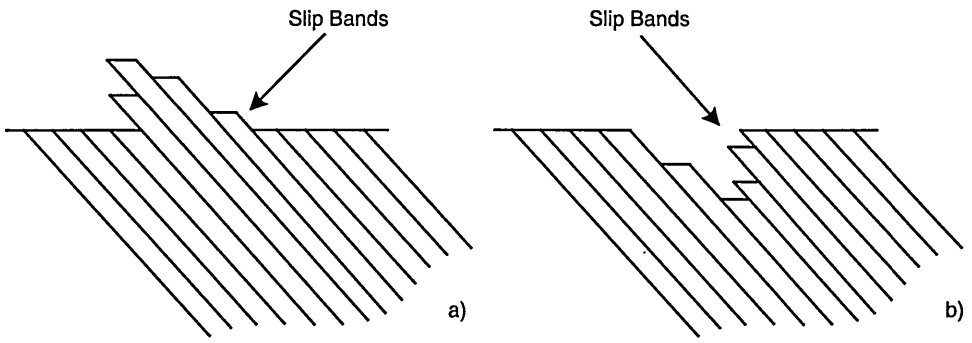


Figure 2: Slip band deformation, a) extrusion, b) intrusion.

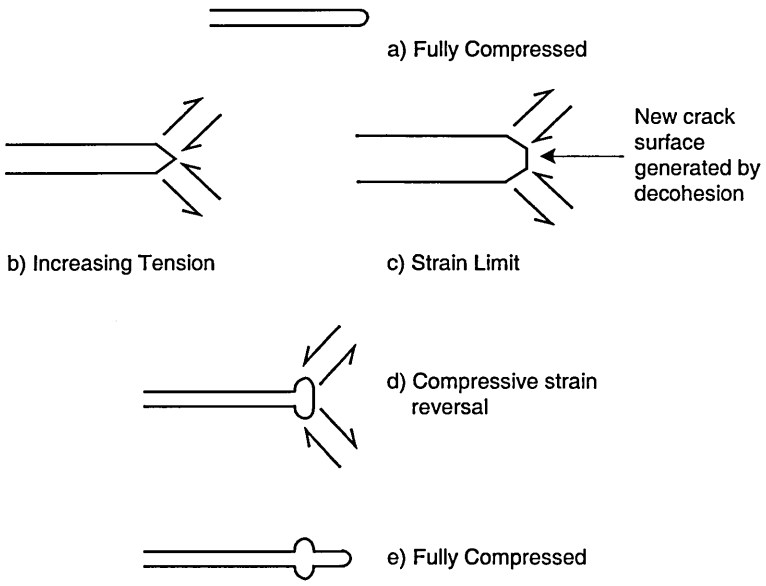


Figure 3: Tomkins' model for Stage II crack propagation.

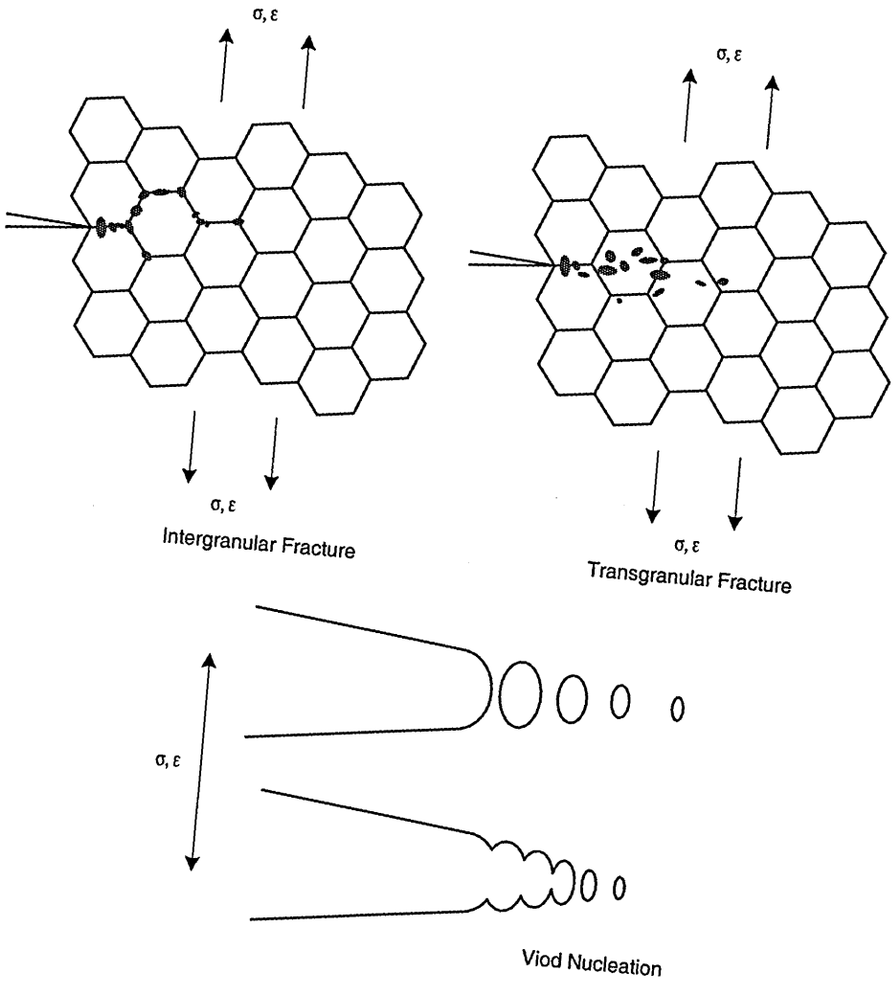


Figure 4: Micromechanisms of fracture.

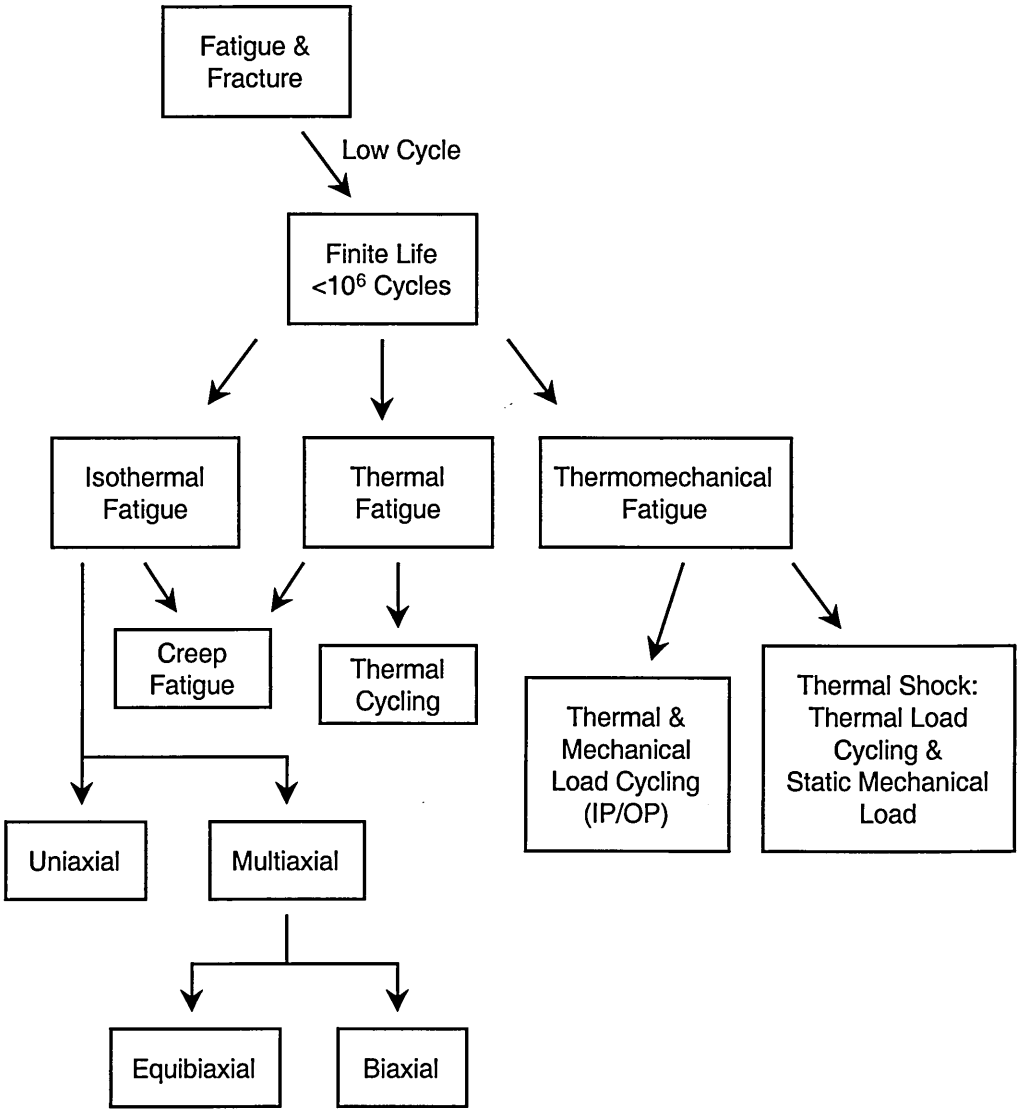


Figure 5: Fatigue and fracture families.

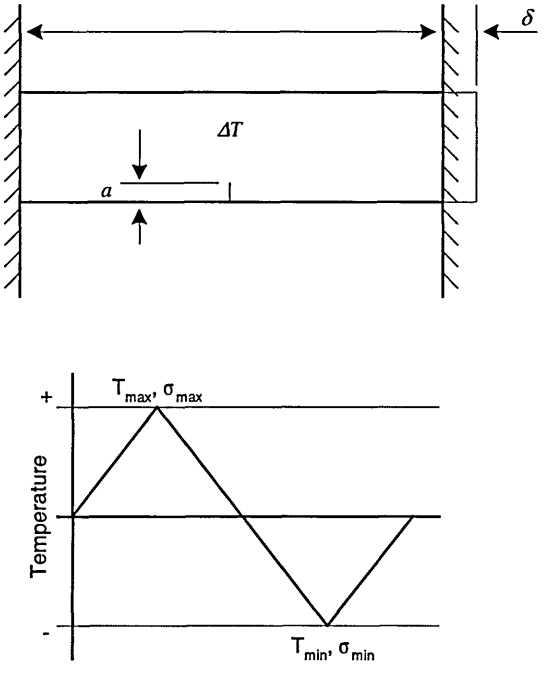


Figure 6: Thermal expansion induced stress.

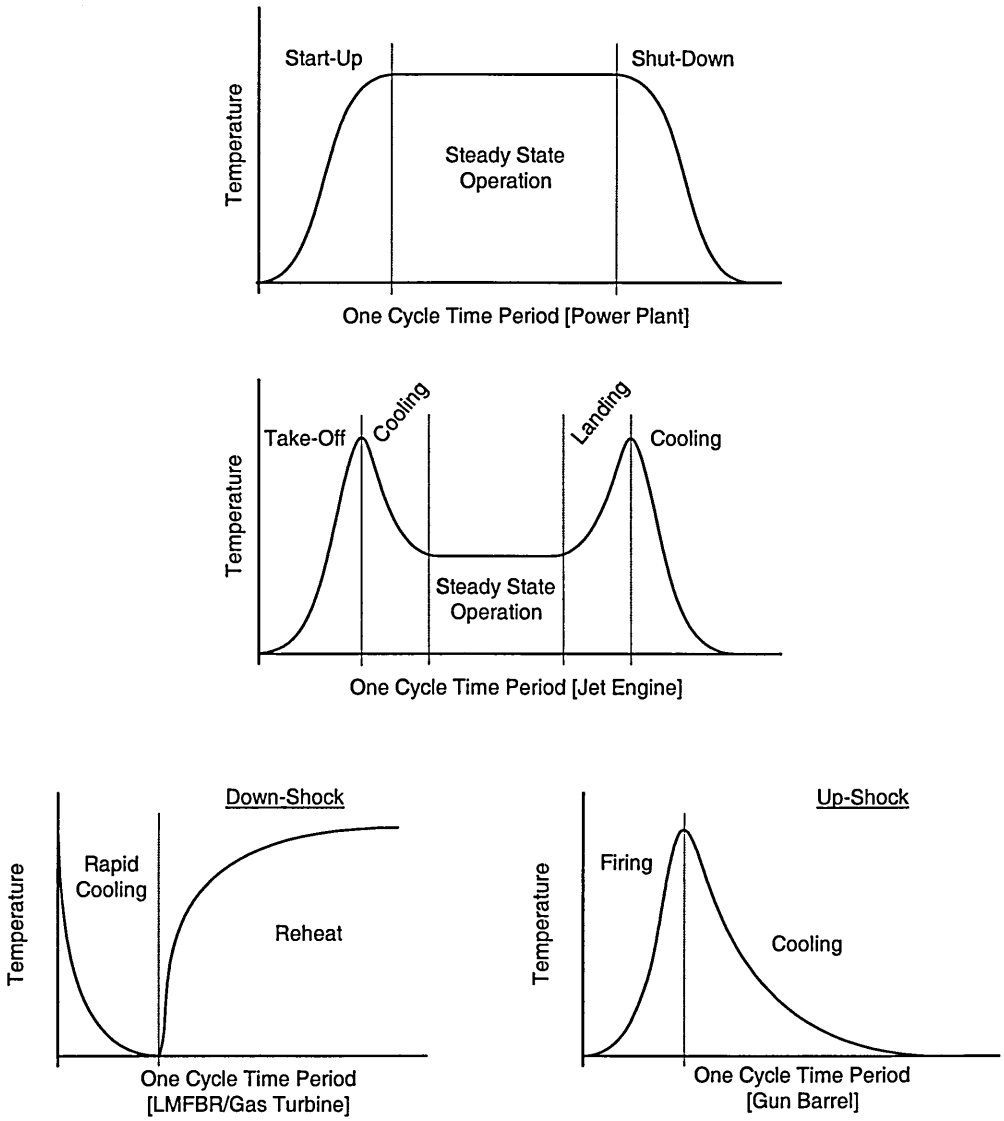


Figure 7: Typical thermal loading cycles.

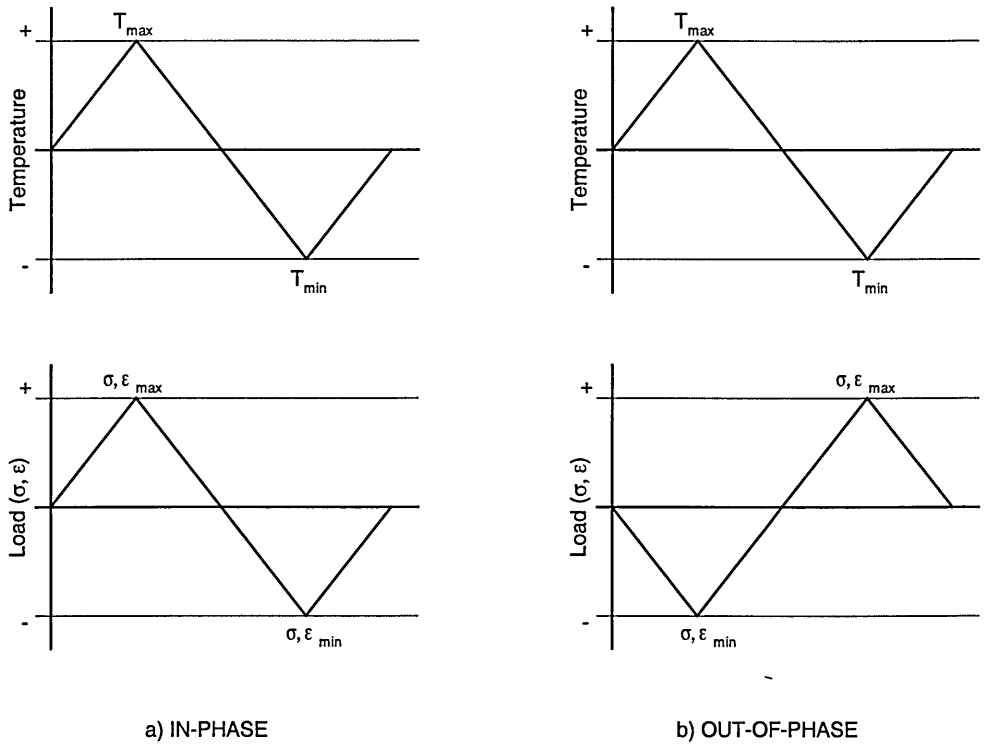
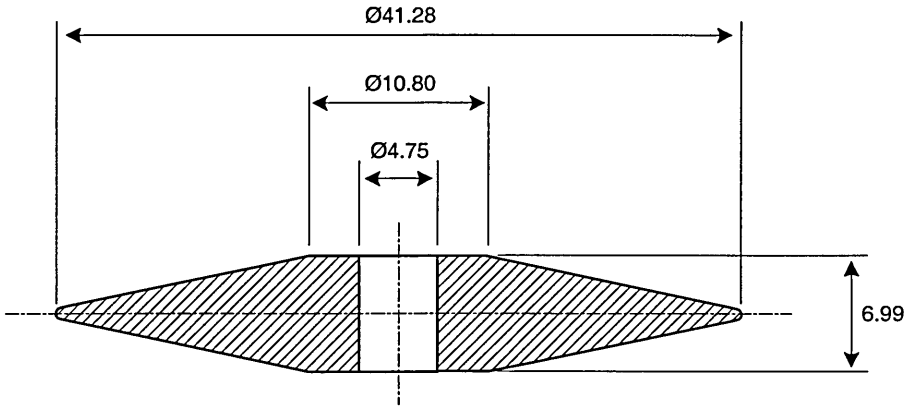


Figure 8: Thermomechanical loading cycles.



Dimensions in millimetres
and approximate to E. Glenny imperial design

Figure 9: Glenny disc specimen geometry.

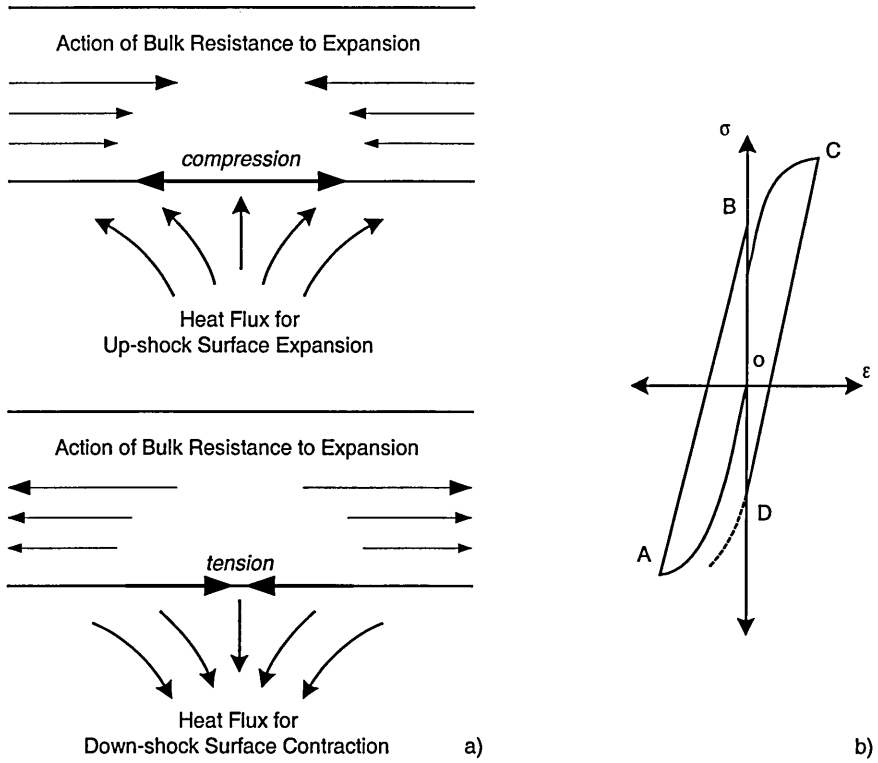


Figure 10: a) Schematic for up and down shock in a thin plate,
b) Hysteresis loop for a thermal up and down shock cycle [66].

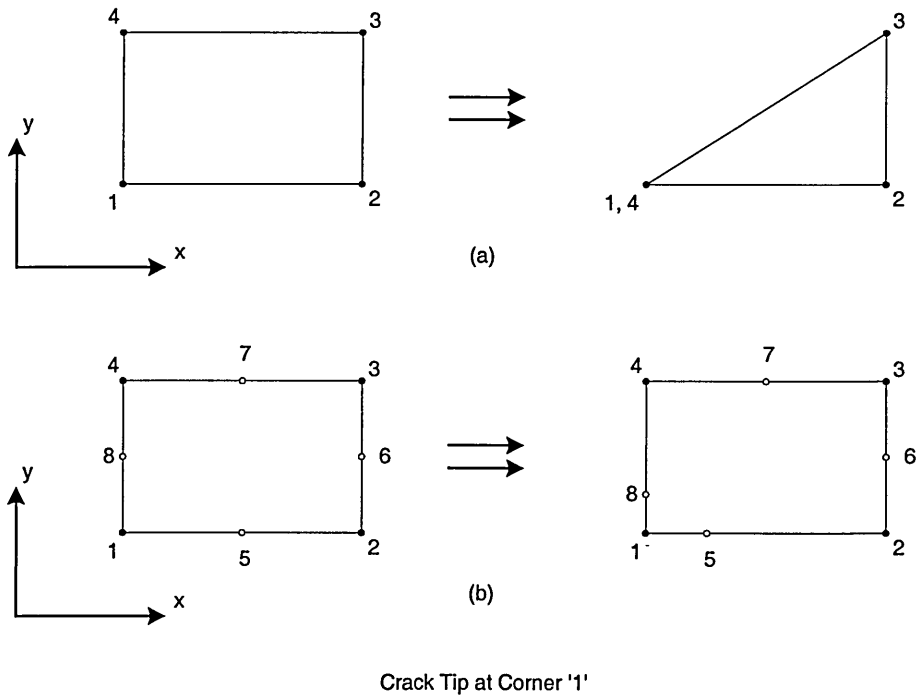


Figure 11: Crack tip finite elements,
a) Tracey's triangular singularity element, b) quarter point element.

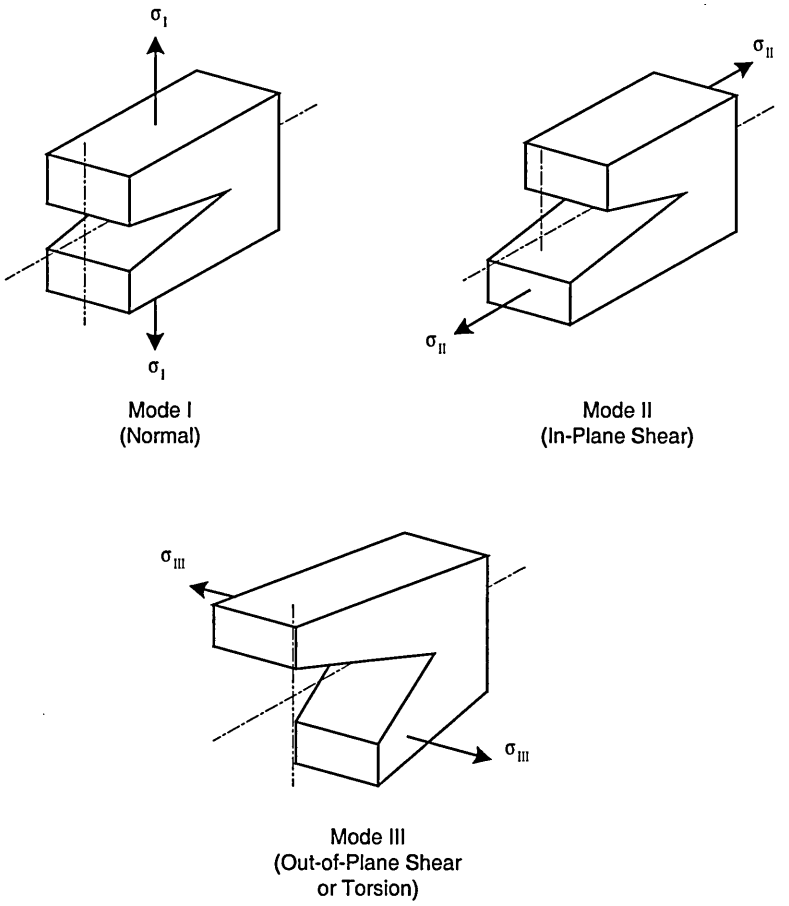


Figure 12: Modes of loading.

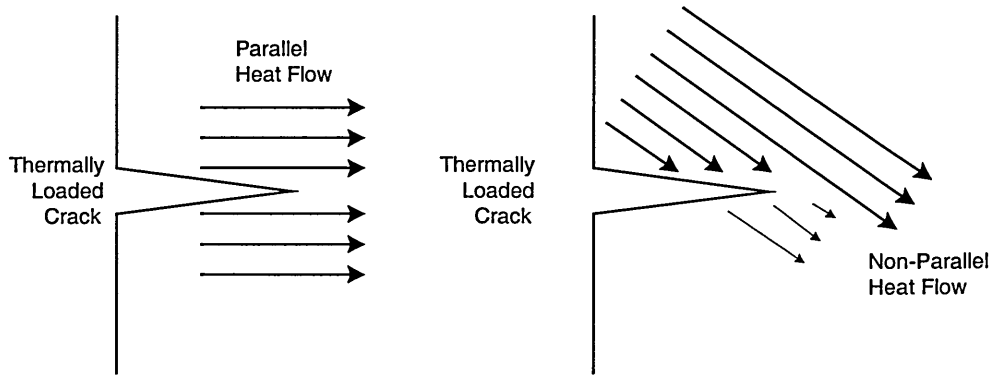


Figure 13: Effect of non-parallel heat flow around a crack tip.

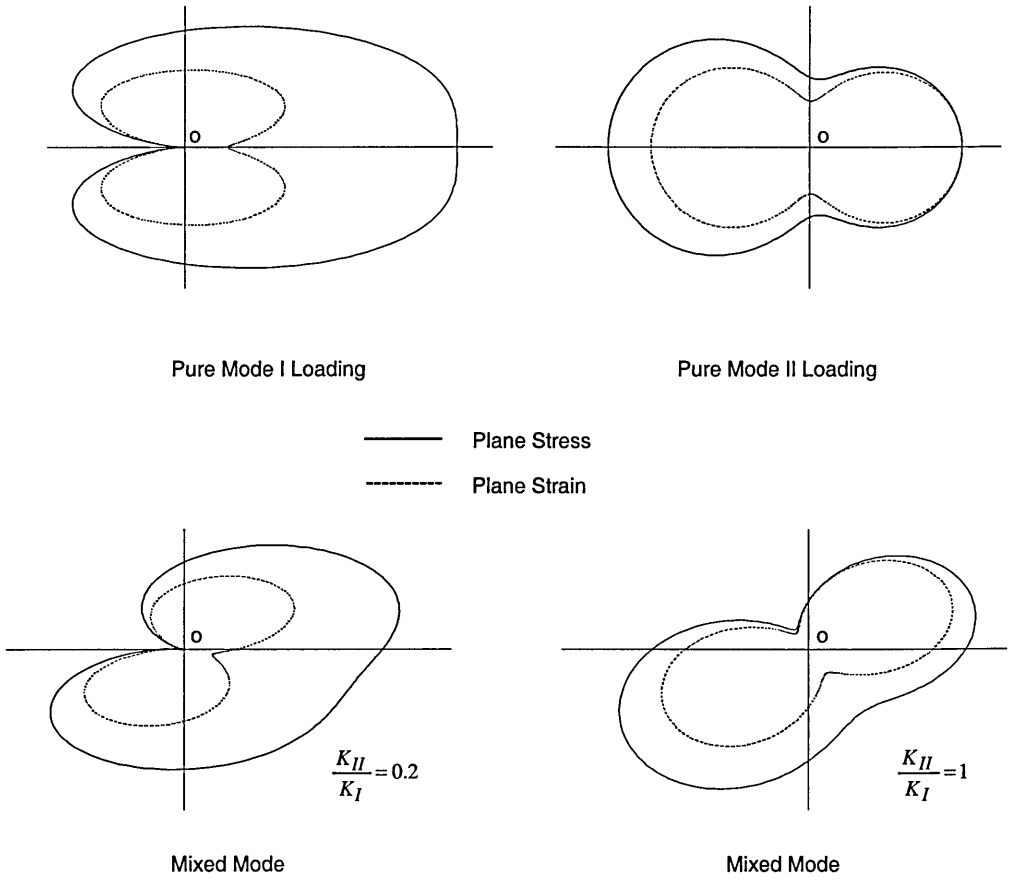


Figure 14: Mixed mode plastic zone shapes, crack tip at '*o*'.

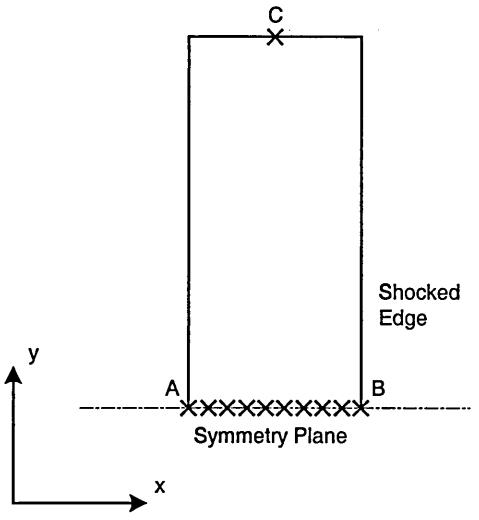


Figure 15: Conceptual thermal shock static stress model.

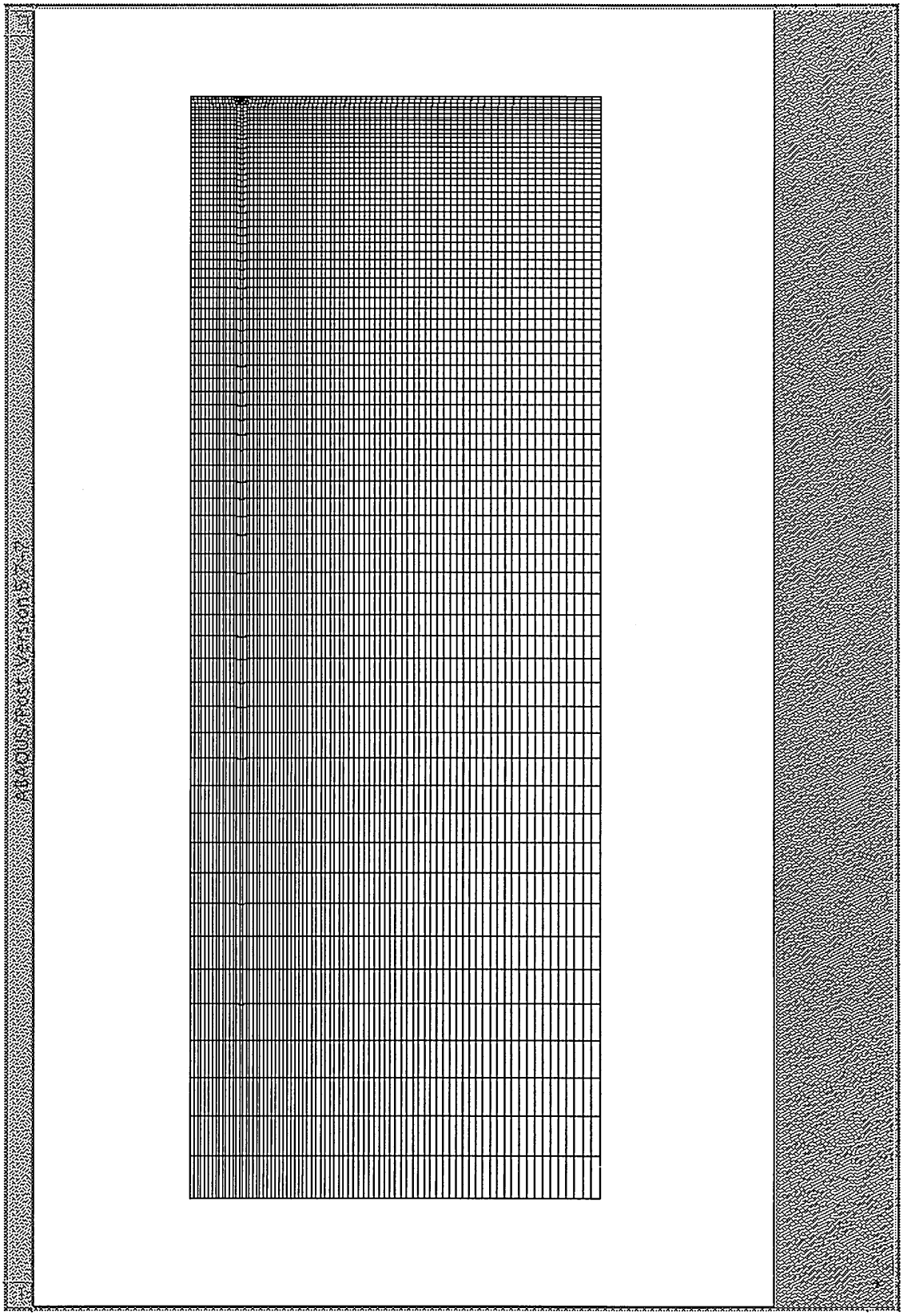


Figure 16a: Thermal shock edge cracked finite element mesh.

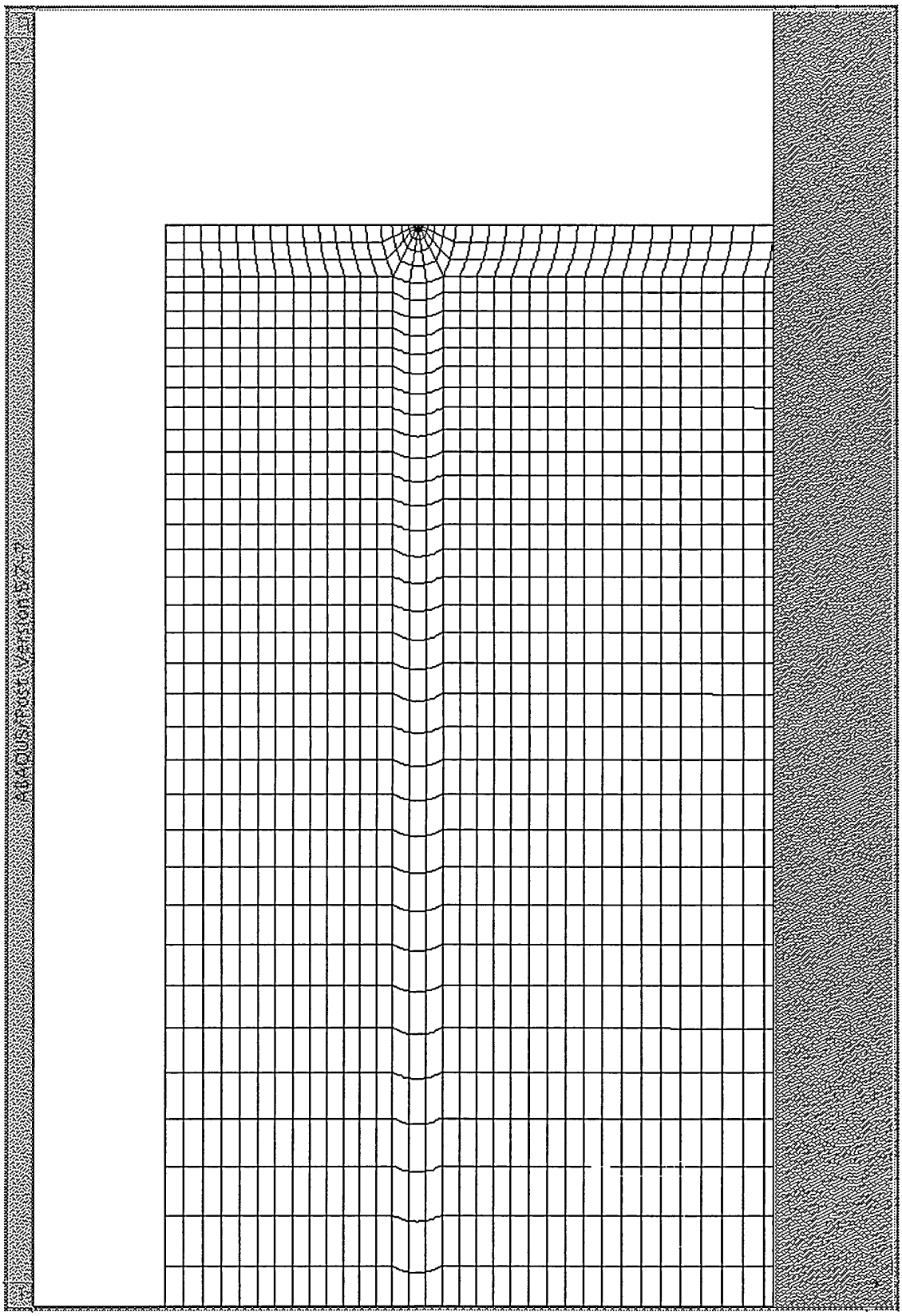


Figure 16b: Thermal shock edge cracked finite element mesh – close up.

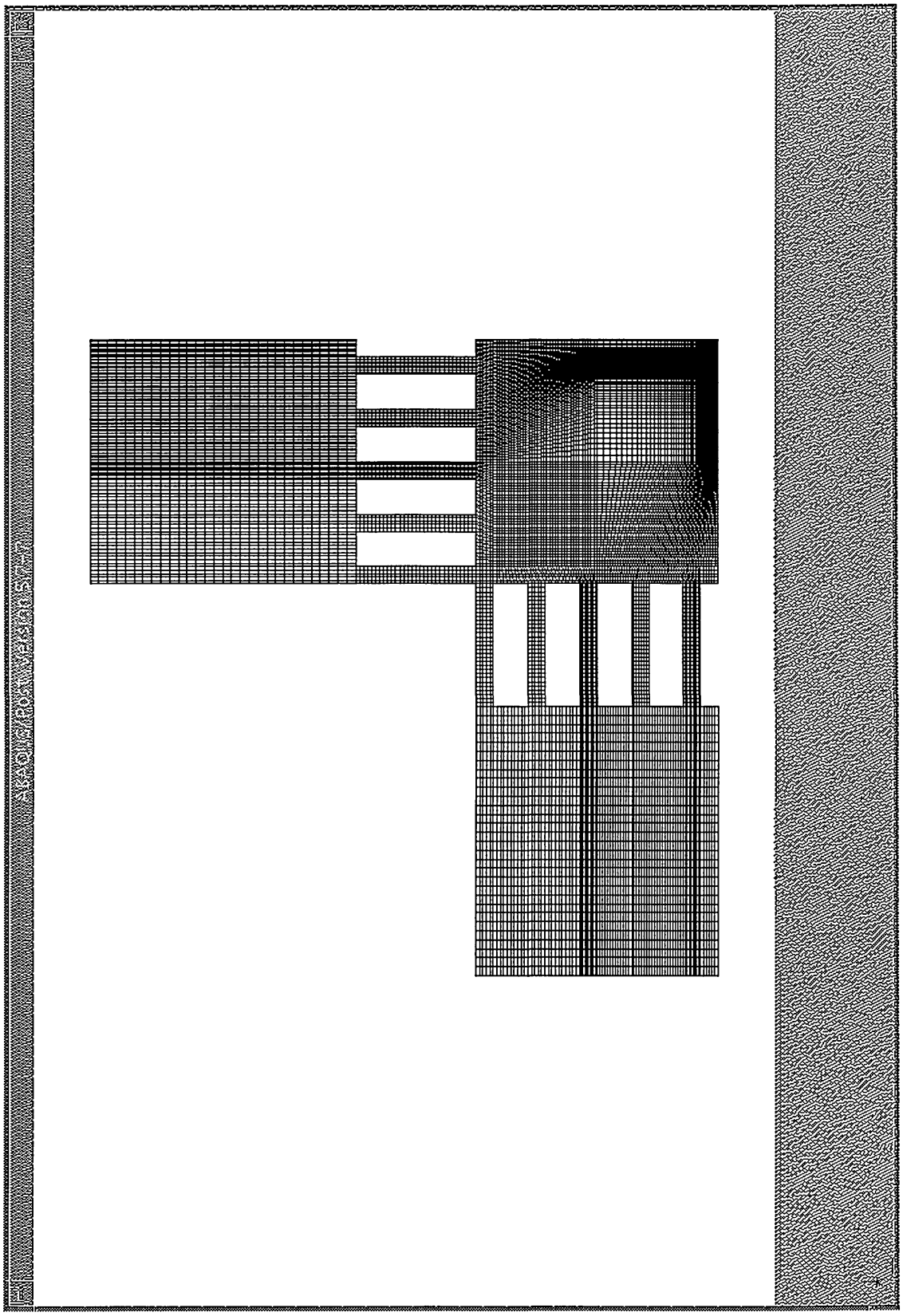


Figure 17a: Cruciform centre cracked finite element mesh.

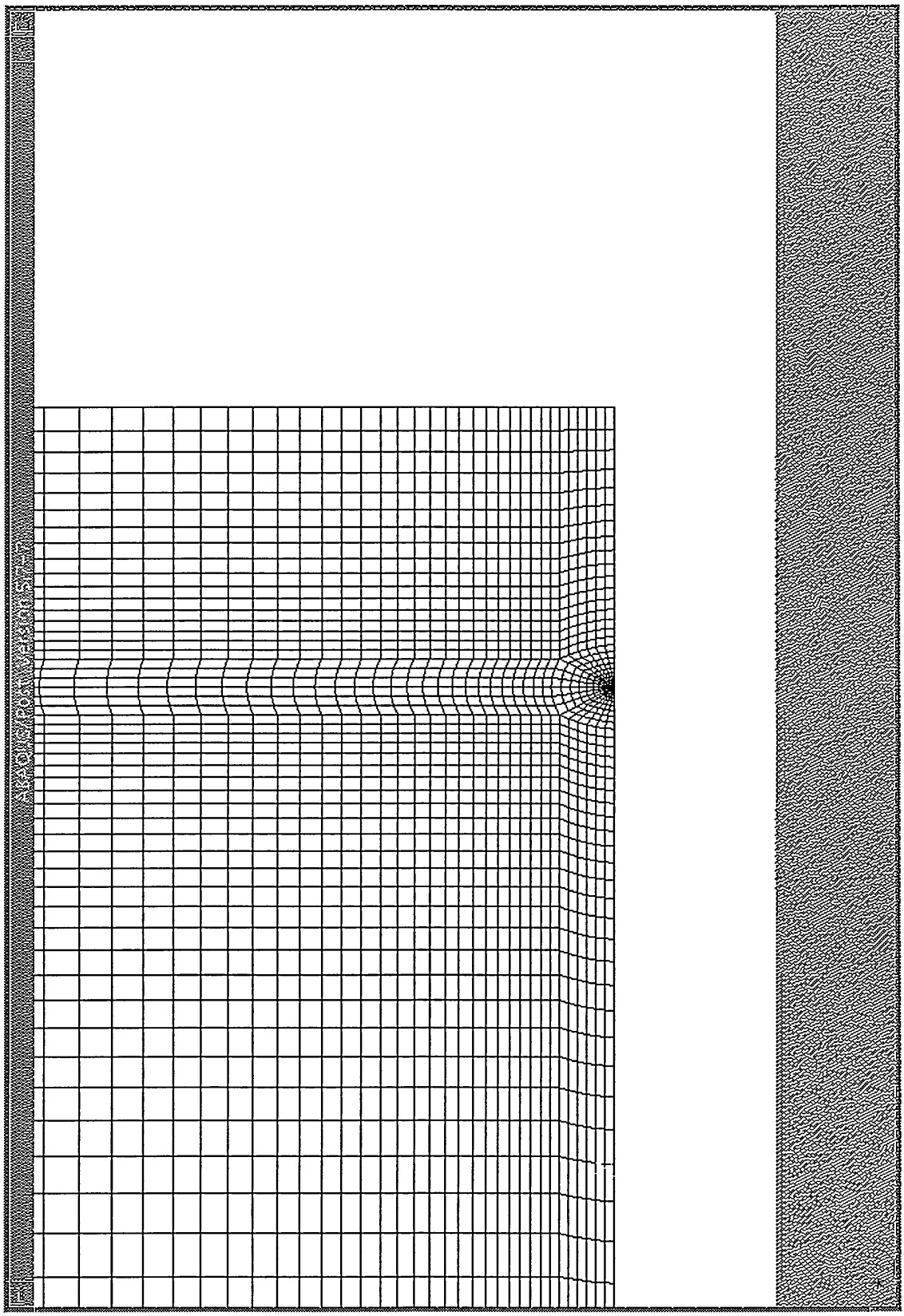


Figure 17b: Cruciform centre cracked finite element mesh – close up.

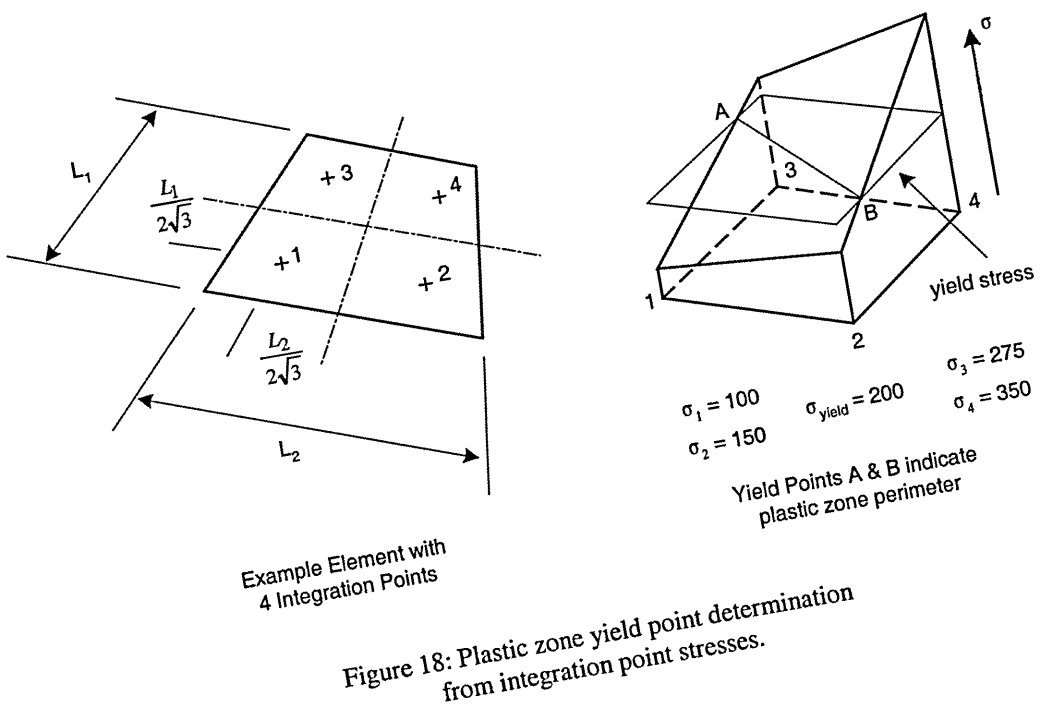


Figure 18: Plastic zone yield point determination from integration point stresses.

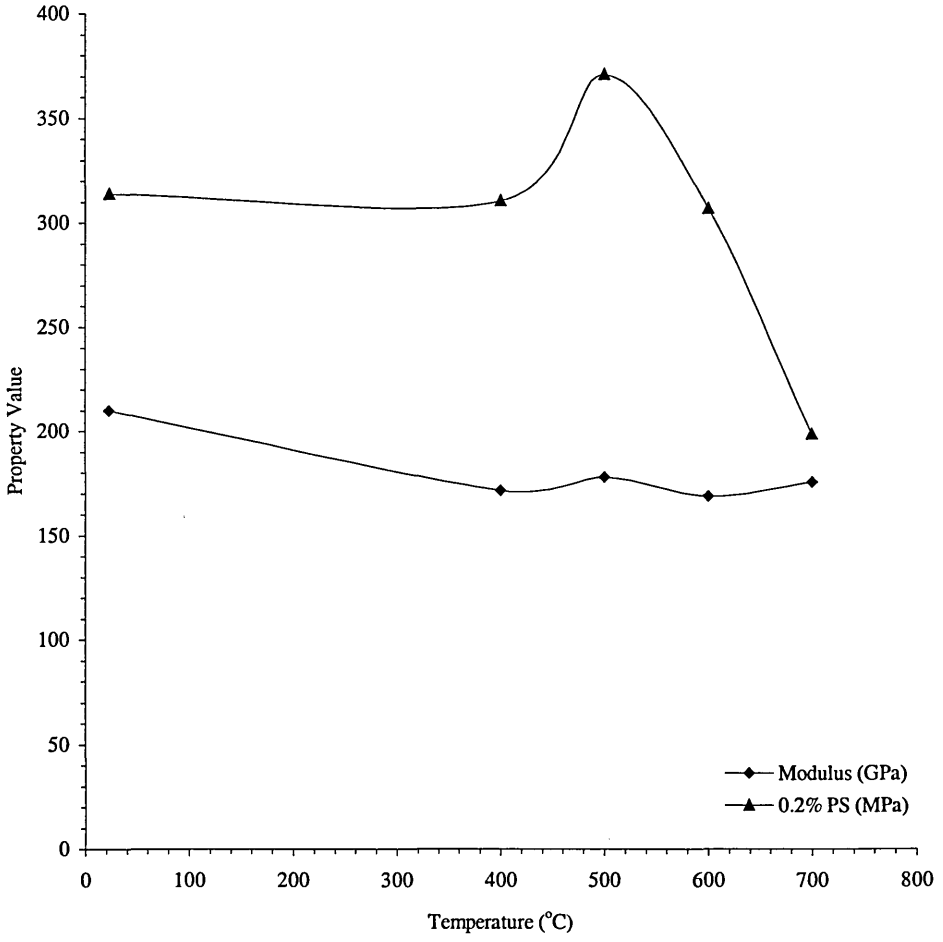


Figure 19: Temperature dependant cyclic modulus and 0.2% proof stress.

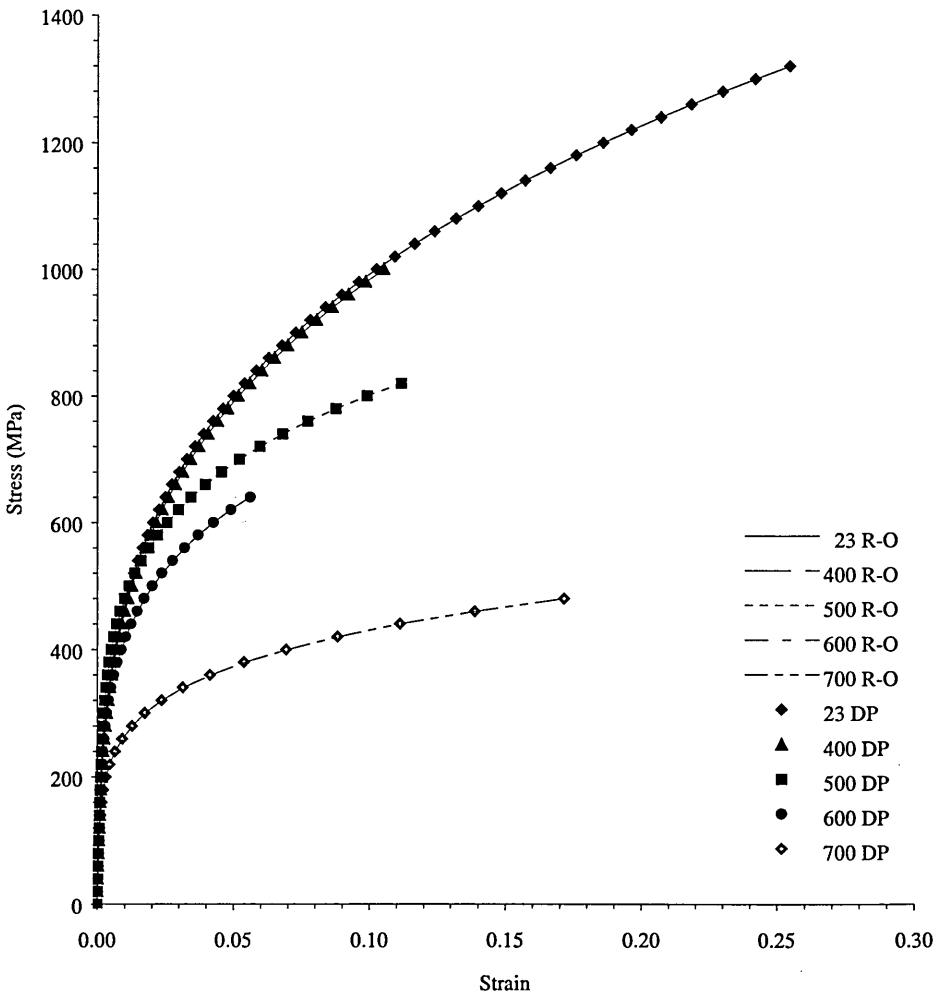


Figure 20: Stress-strain response curves by Ramberg-Osgood relationship.

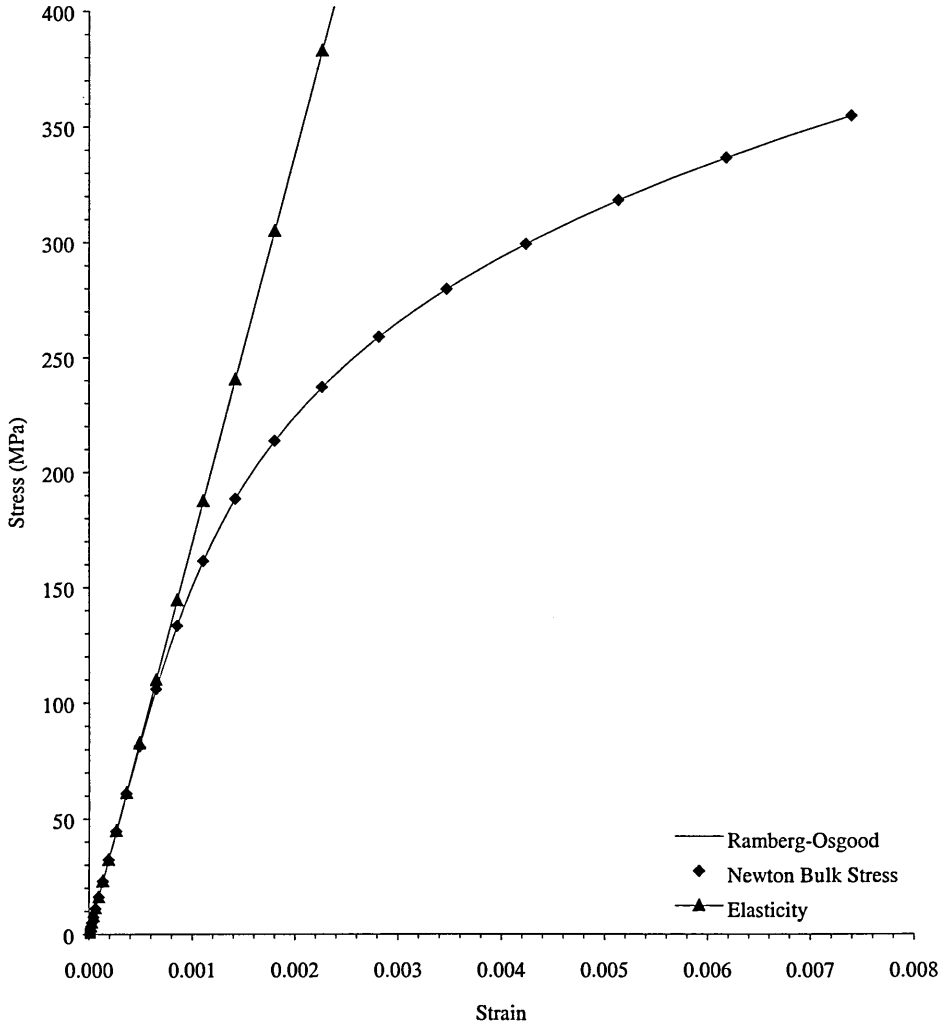


Figure 21: Elastic-Plastic Ramberg-Osgood & cyclic elasticity material models.

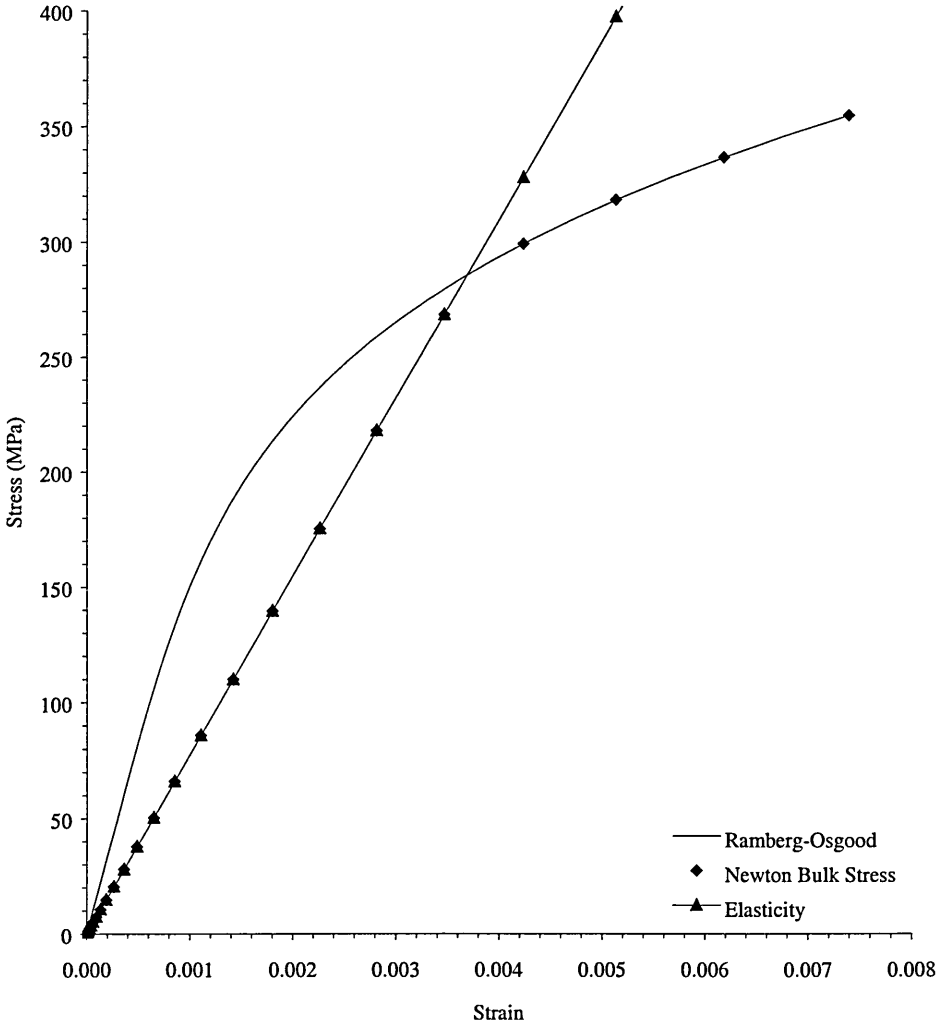


Figure 22: Linear-Elastic Elastic-Plastic material model using pseudo mechanical modulus and Ramberg-Osgood plasticity.

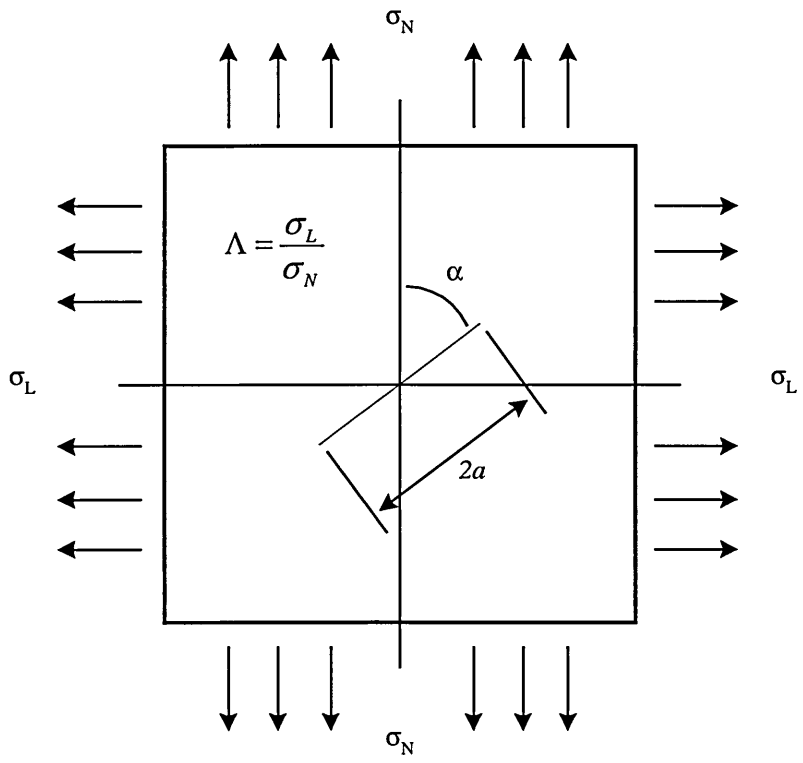


Figure 23: Non-singular stress description for biaxial loading.

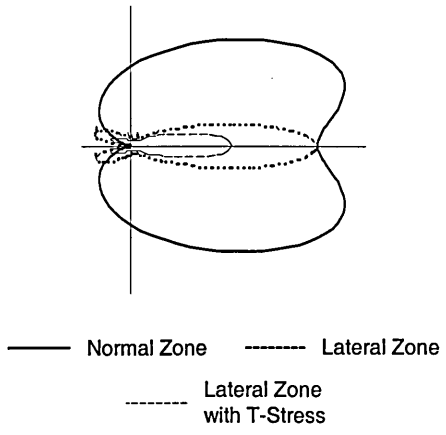


Figure 24: Normal and lateral plastic zones.

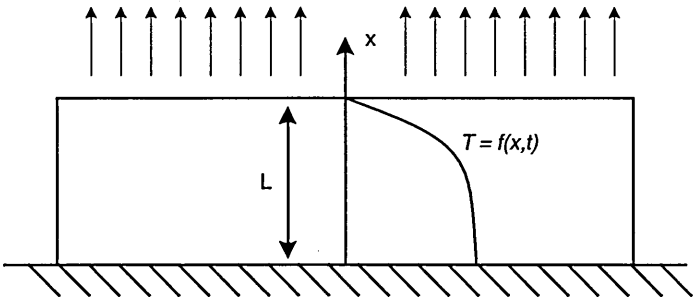
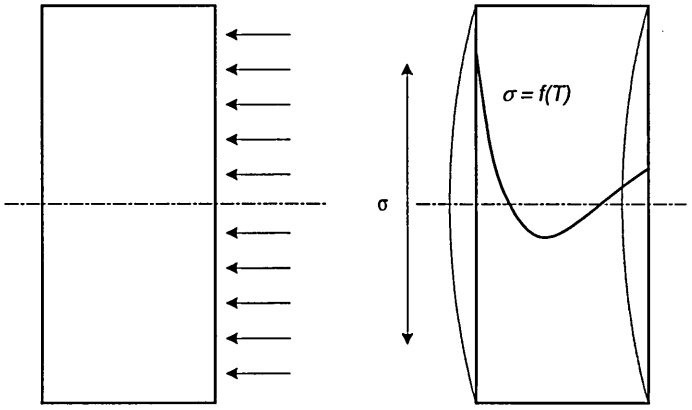
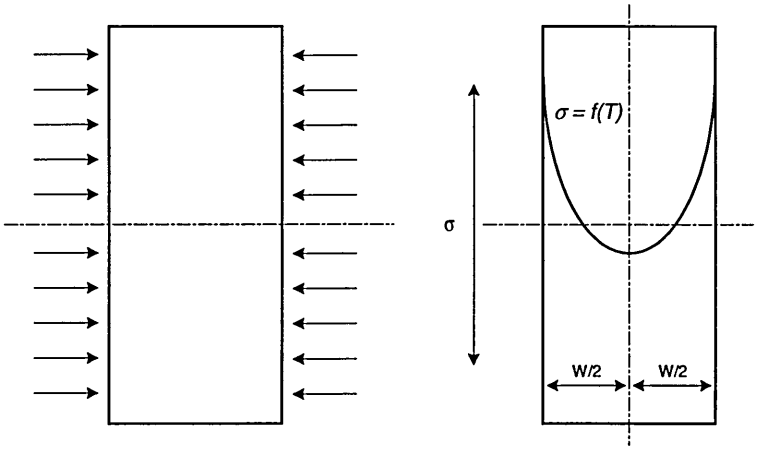


Figure 25: One dimensional cooled edge model
with opposite edge perfectly insulated.



a) Single edge cooling



b) Symmetric double edge cooling

Figure 26: Typical stress profiles in symmetric and non-symmetric cooling.

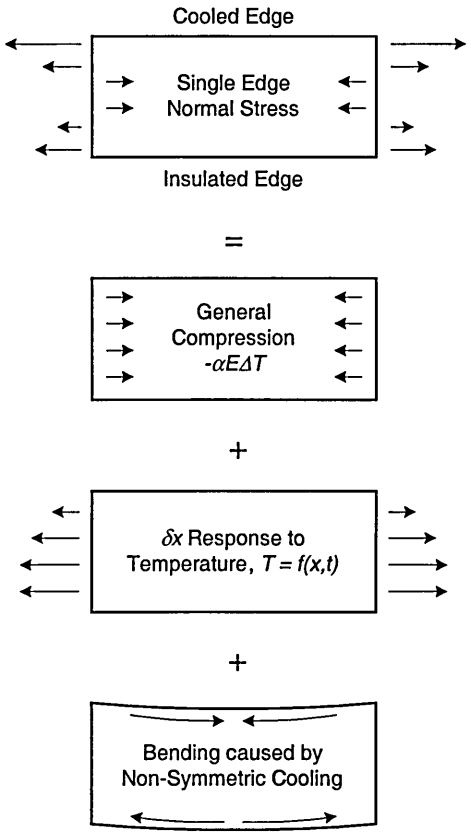


Figure 27: Superposition of thermal stress effects.

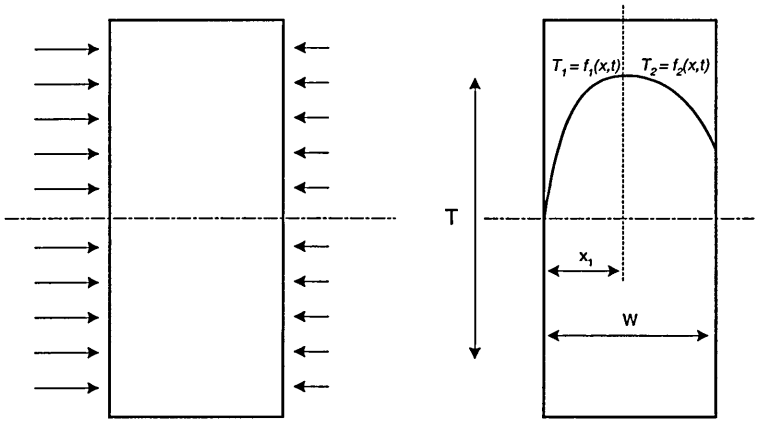


Figure 28: Dual thermal distribution equations for non-symmetric cooling.

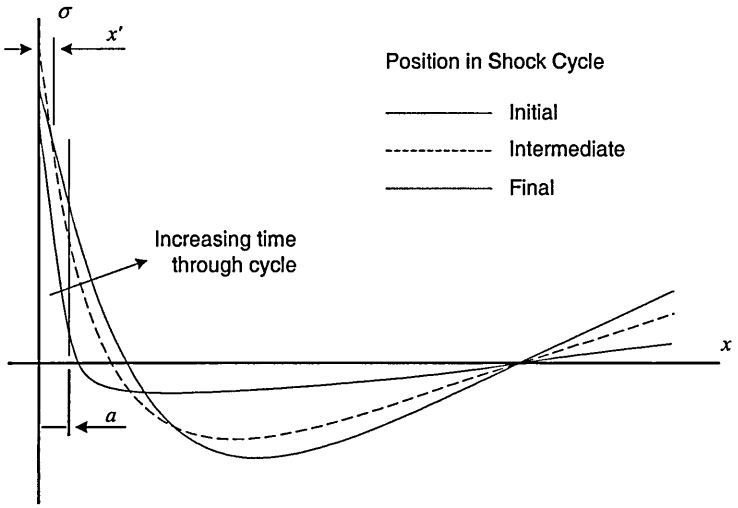


Figure 29: Thermal shock stresses through cycle.

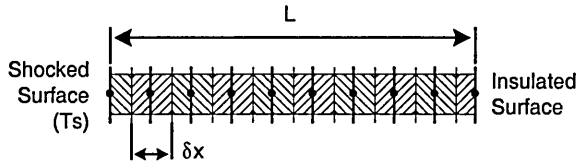


Figure 30: 1D finite difference model,
perfectly insulated edge and rapidly cooled opposing edge.

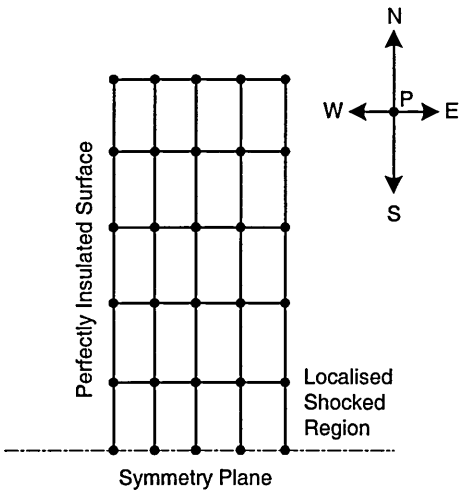


Figure 31: 2D finite difference model with localised edge shock.

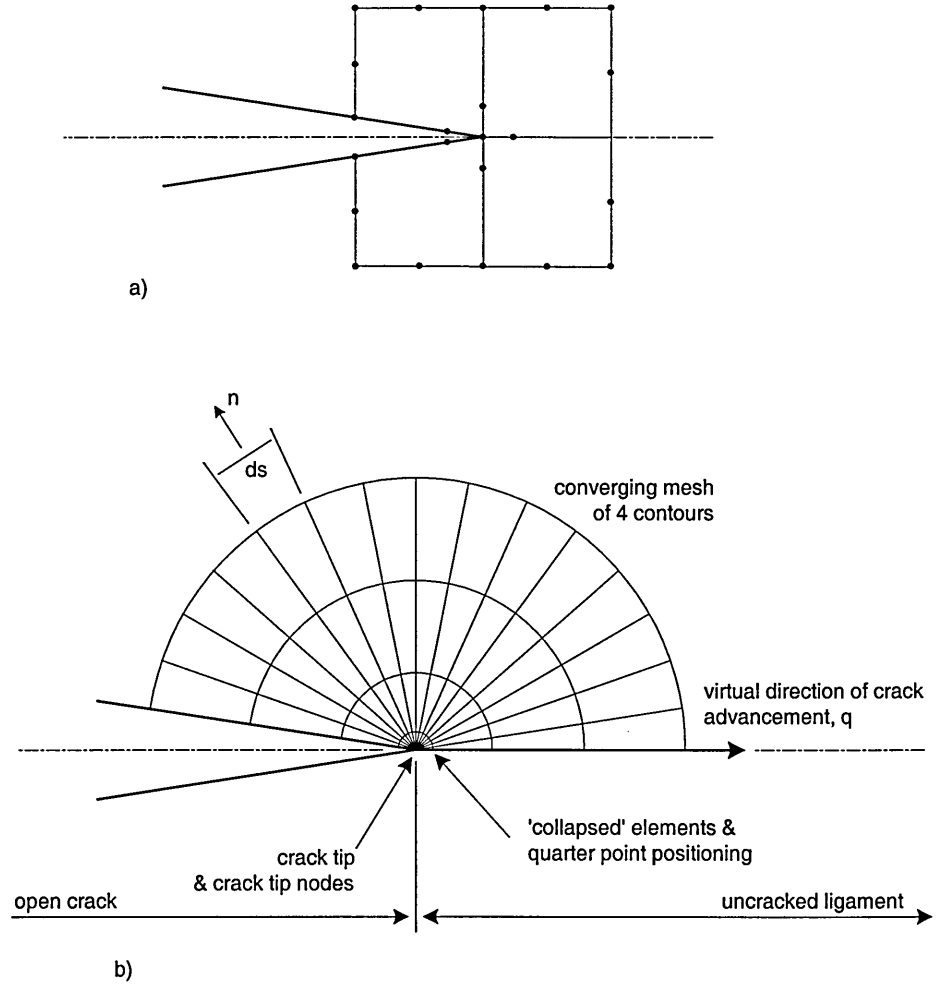


Figure 32: a) Henshall & Shaw quarter point arrangement, b) Contemporary arrangement.

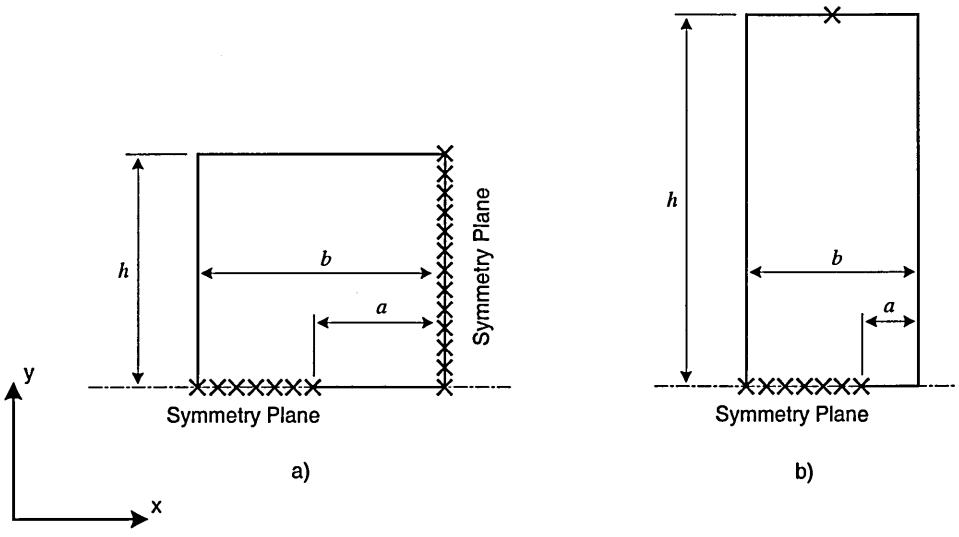


Figure 33: Model constructs for fracture mechanics verification.

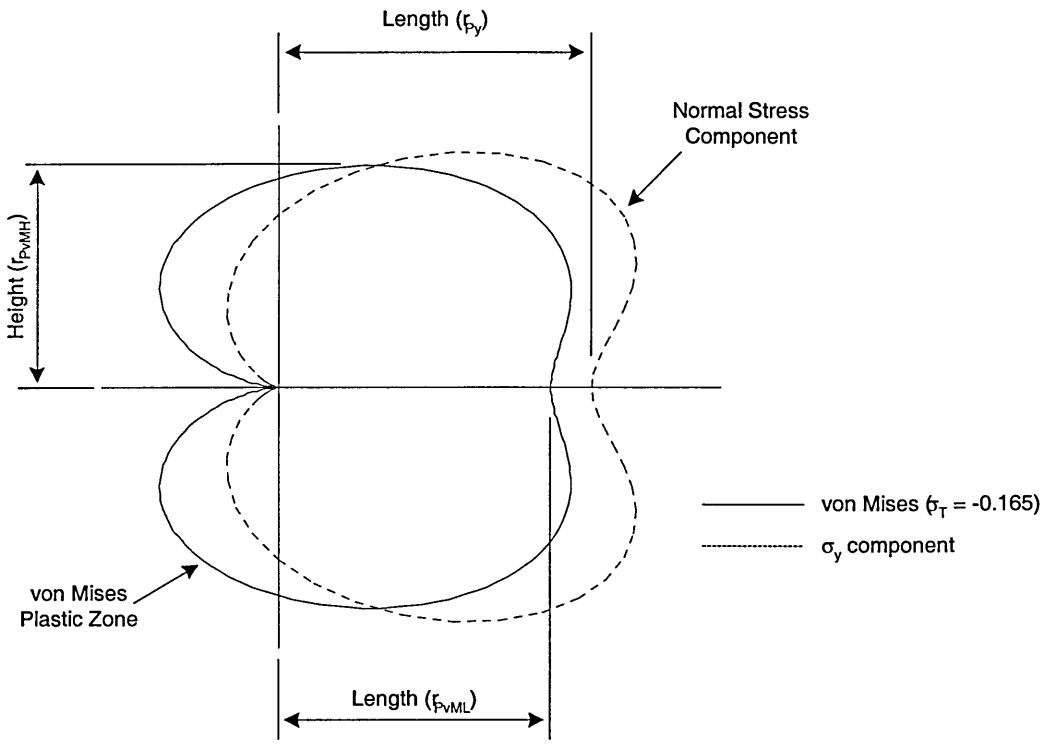


Figure 34: Determination of von Mises plastic zone parameter.

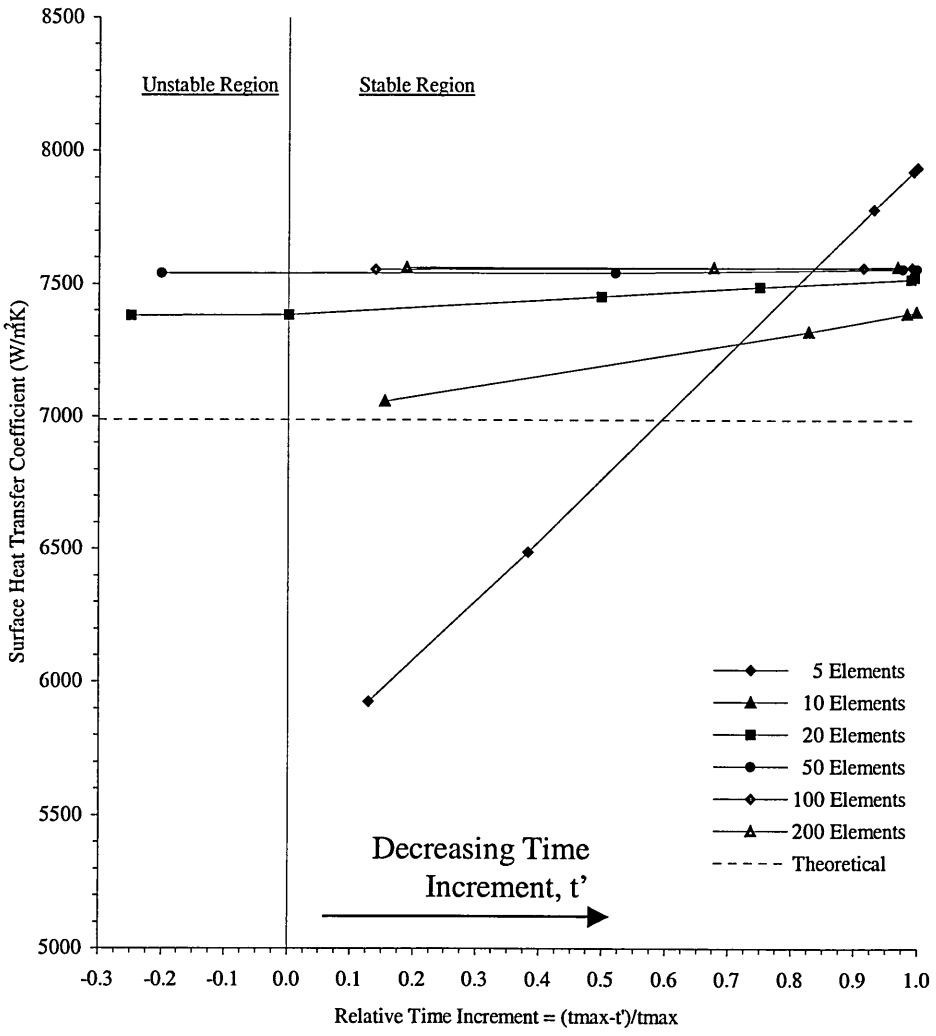


Figure 35: Relative time increment to forced convection refinement for 625-225°C in 3s.

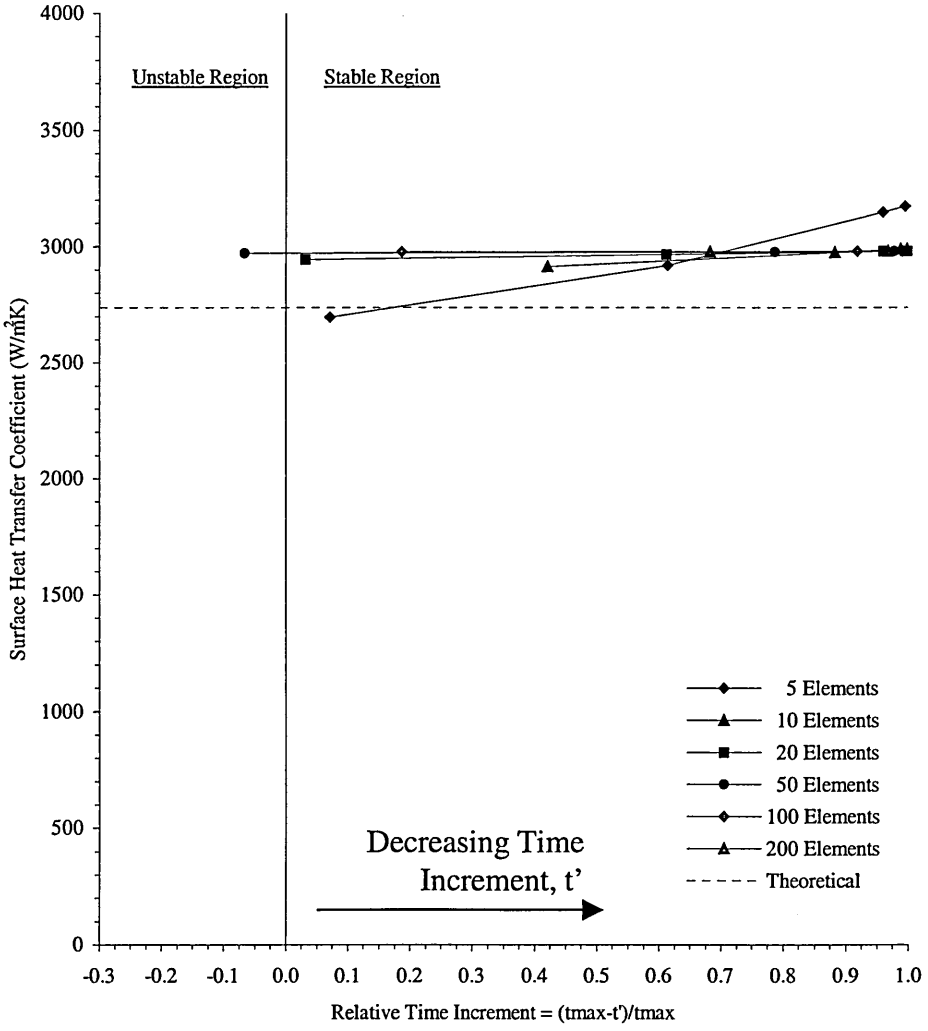


Figure 36: Relative time increment to forced convection refinement for 650-350°C in 5s.

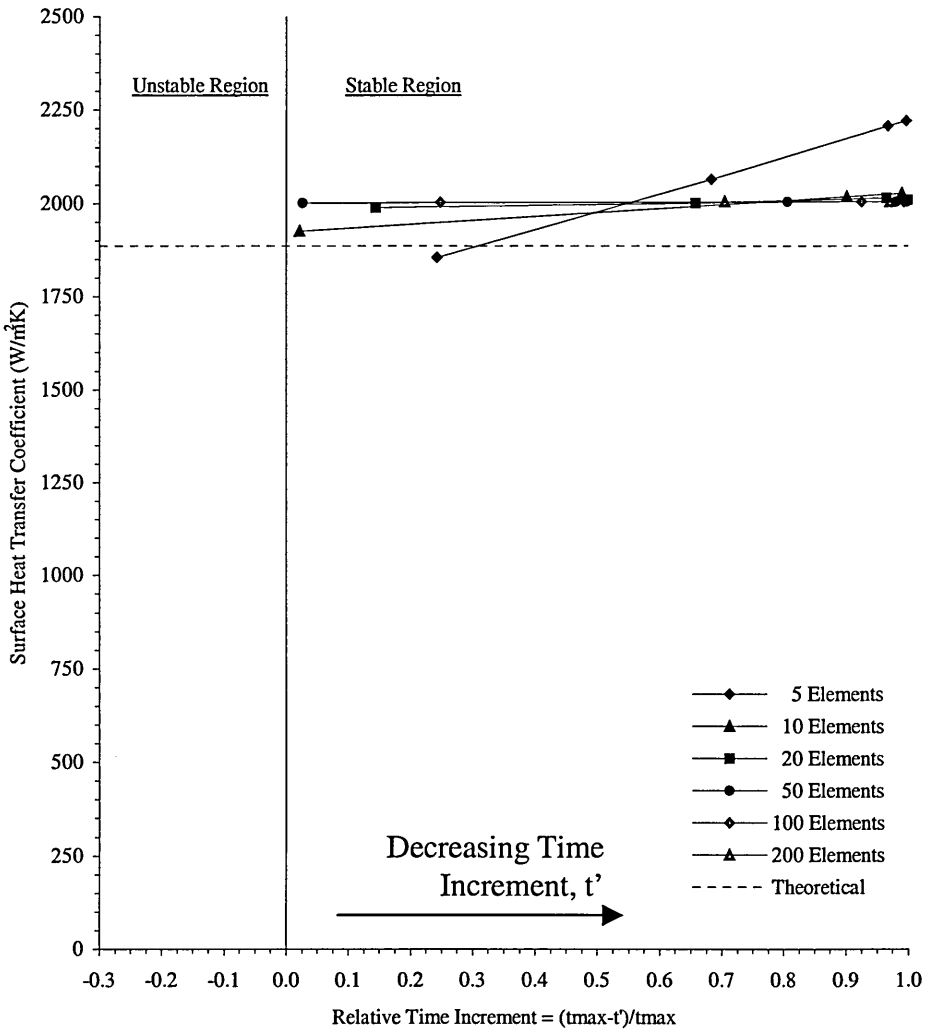


Figure 37: Relative time increment to forced convection refinement for 550-350°C in 5s.

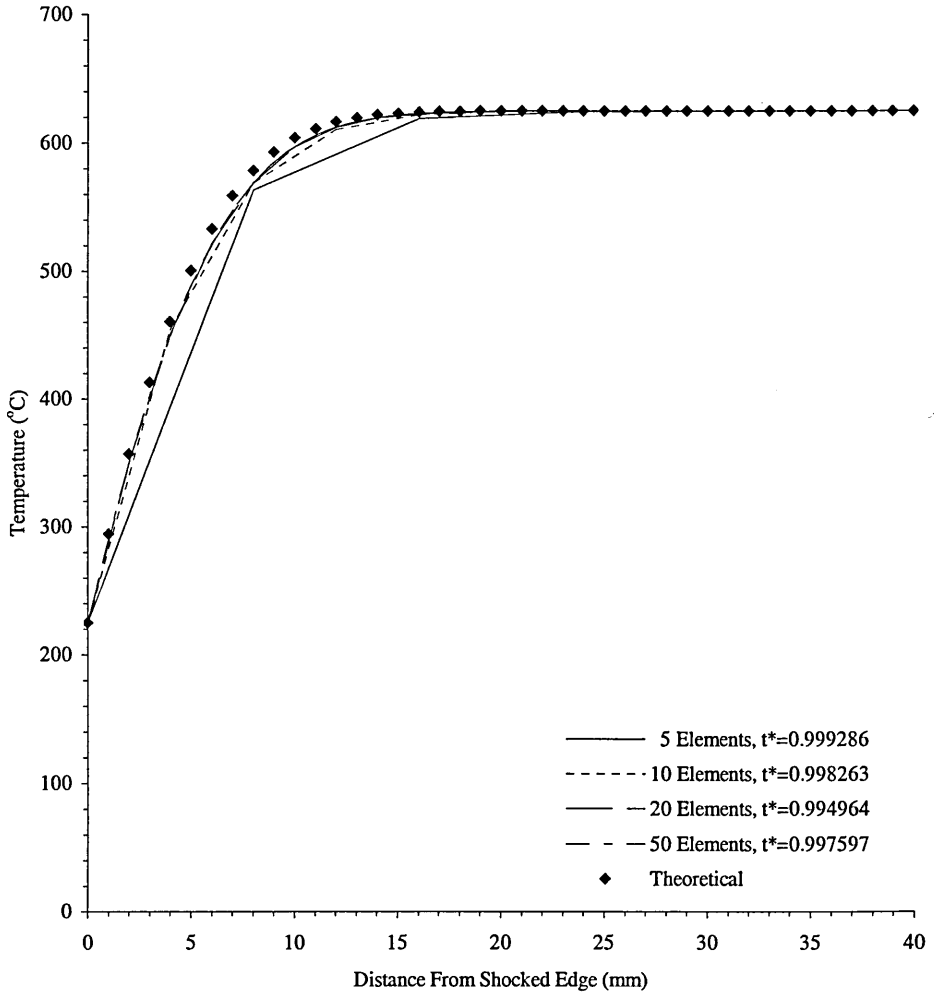


Figure 38: Thermal distribution refinement and theoretical eigenvalue solution for 625-225°C in 3s.

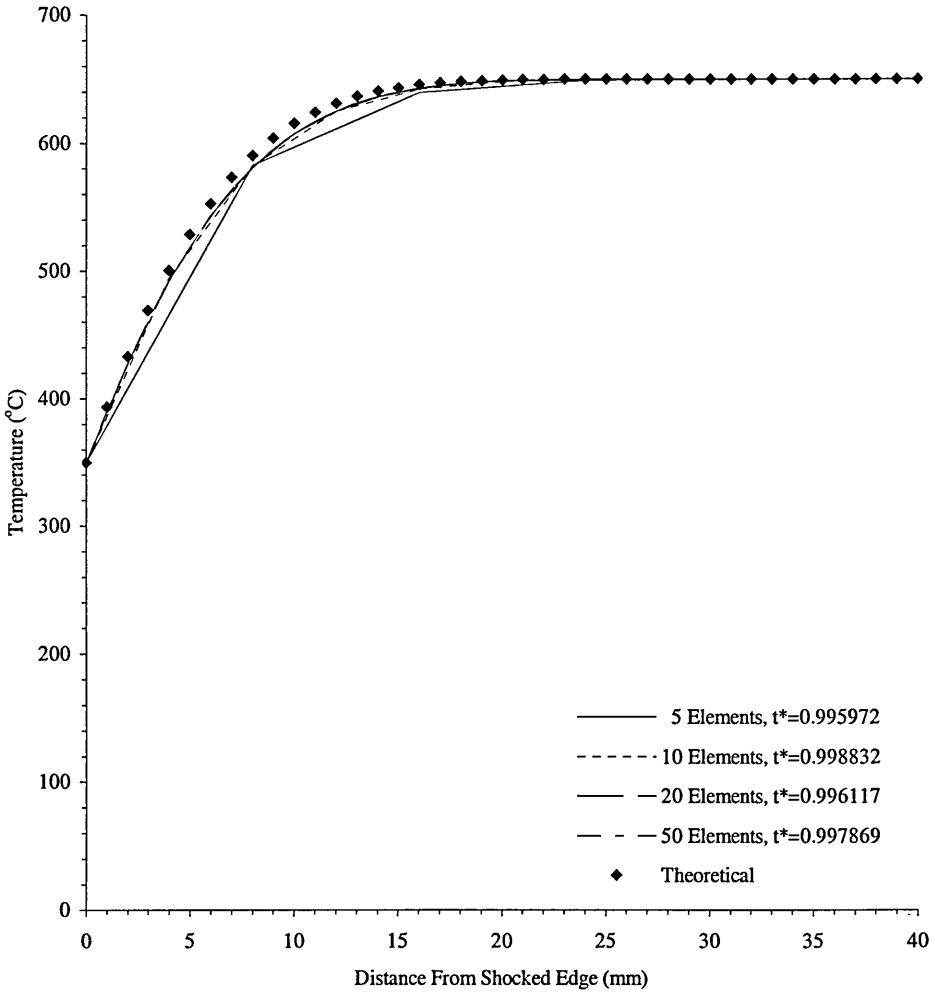


Figure 39: Thermal distribution refinement and theoretical eigenvalue solution for 650-350°C in 5s.

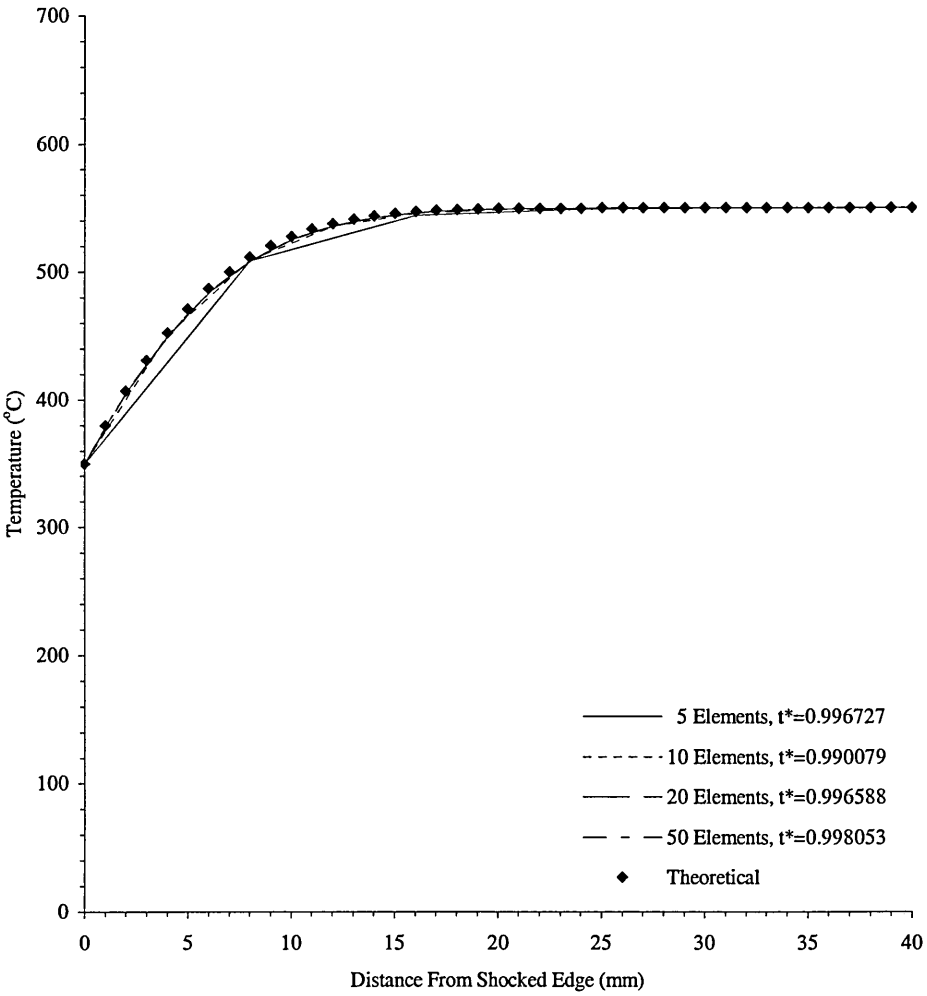


Figure 40: Thermal distribution refinement and theoretical eigenvalue solution for 550-350°C in 5s.

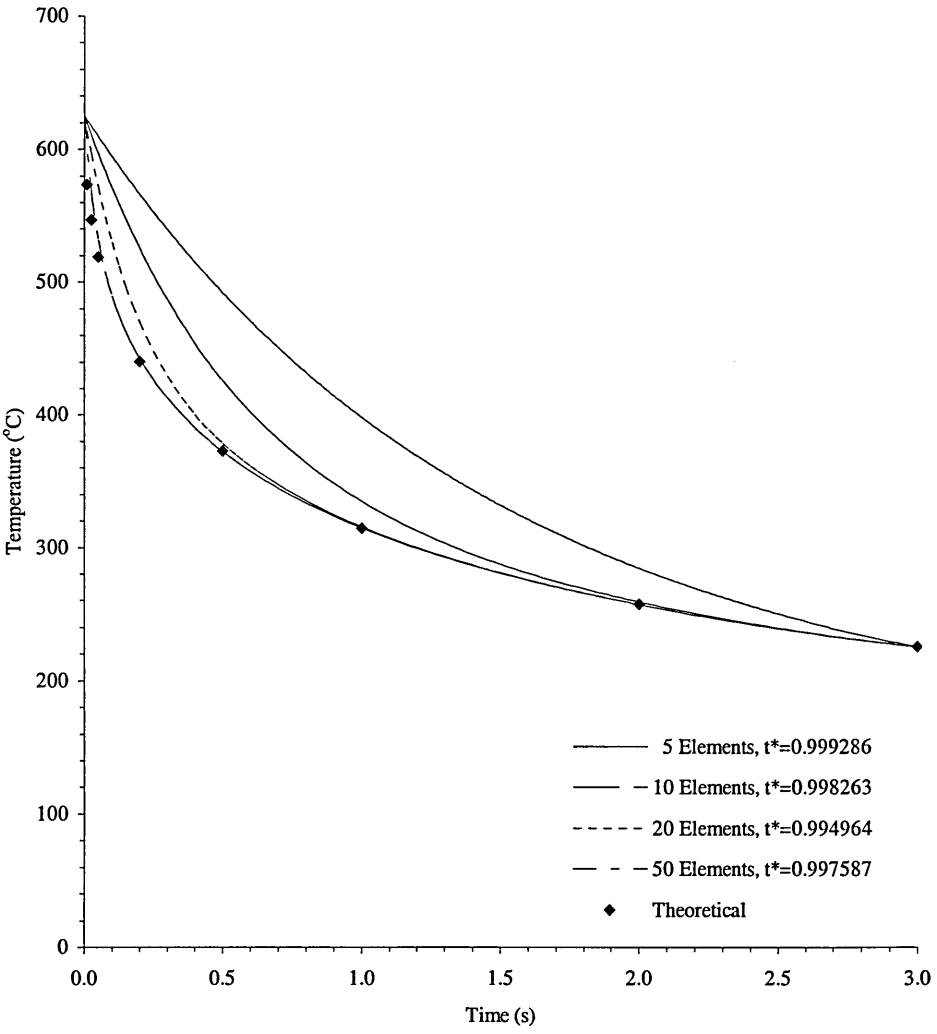


Figure 41: Cooling curve refinement for 625-225°C in 3s.

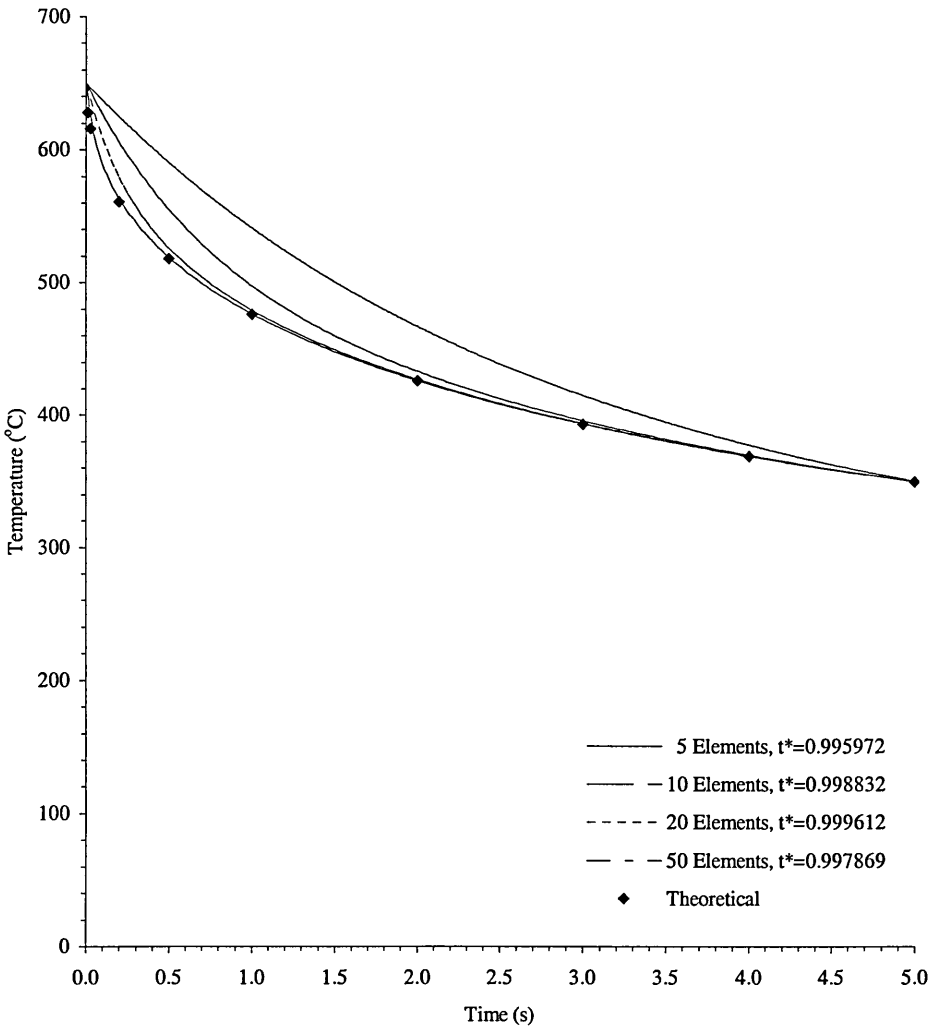


Figure 42: Cooling curve refinement for 650-350°C in 5s.

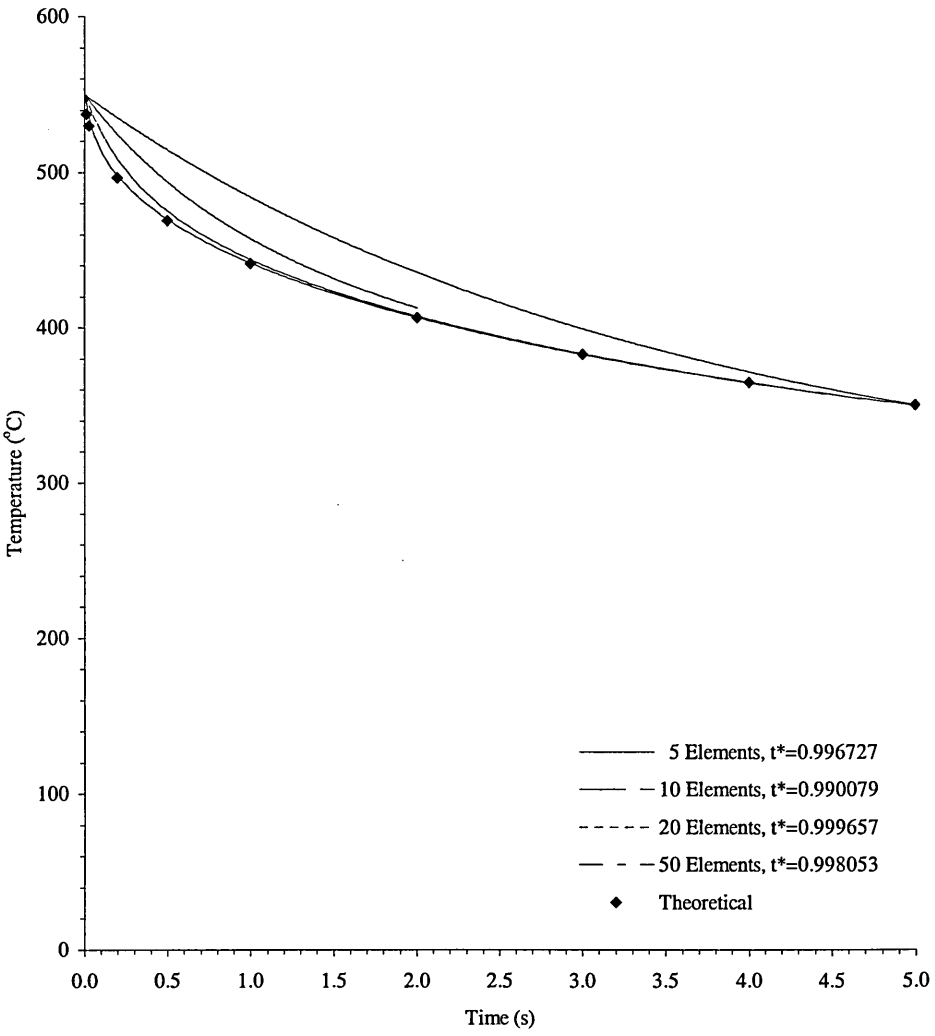


Figure 43: Cooling curve refinement for 650-350°C in 5s.

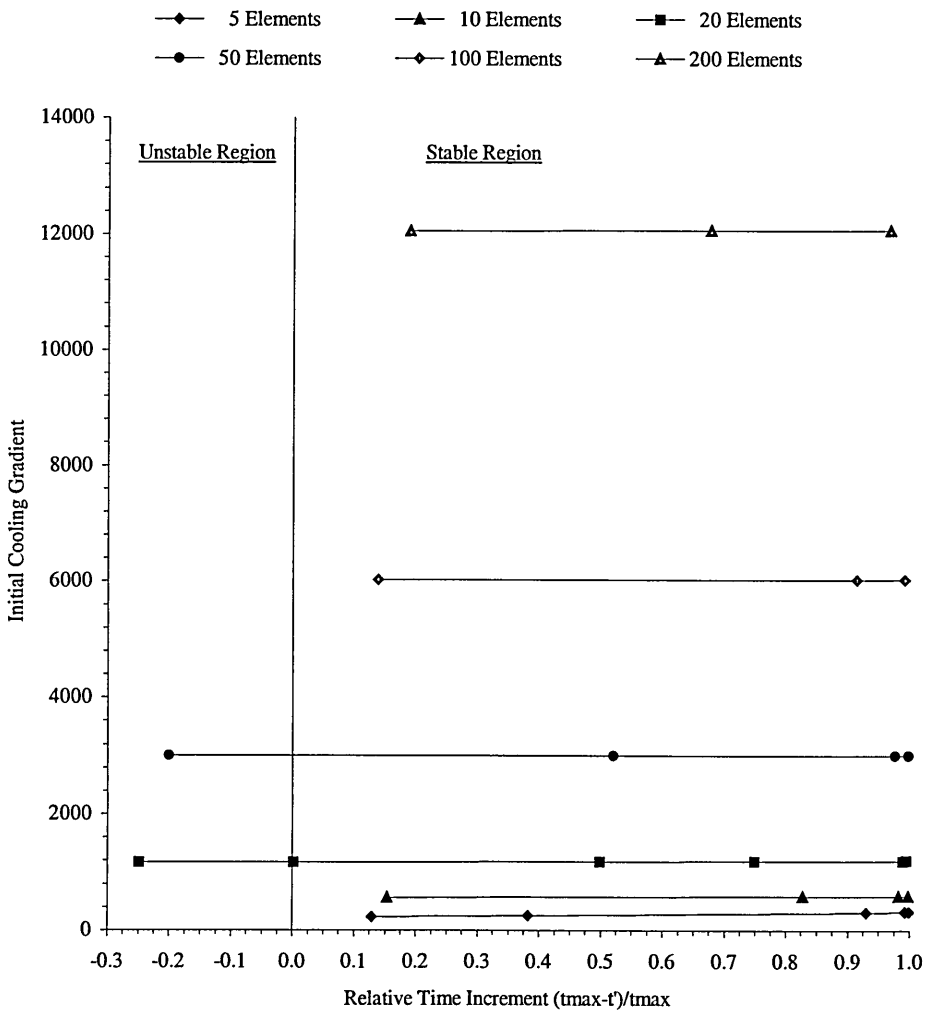


Figure 44: Relative time increment to initial cooling rate (dT/dt at t = 0s) for 625-225°C in 3s.

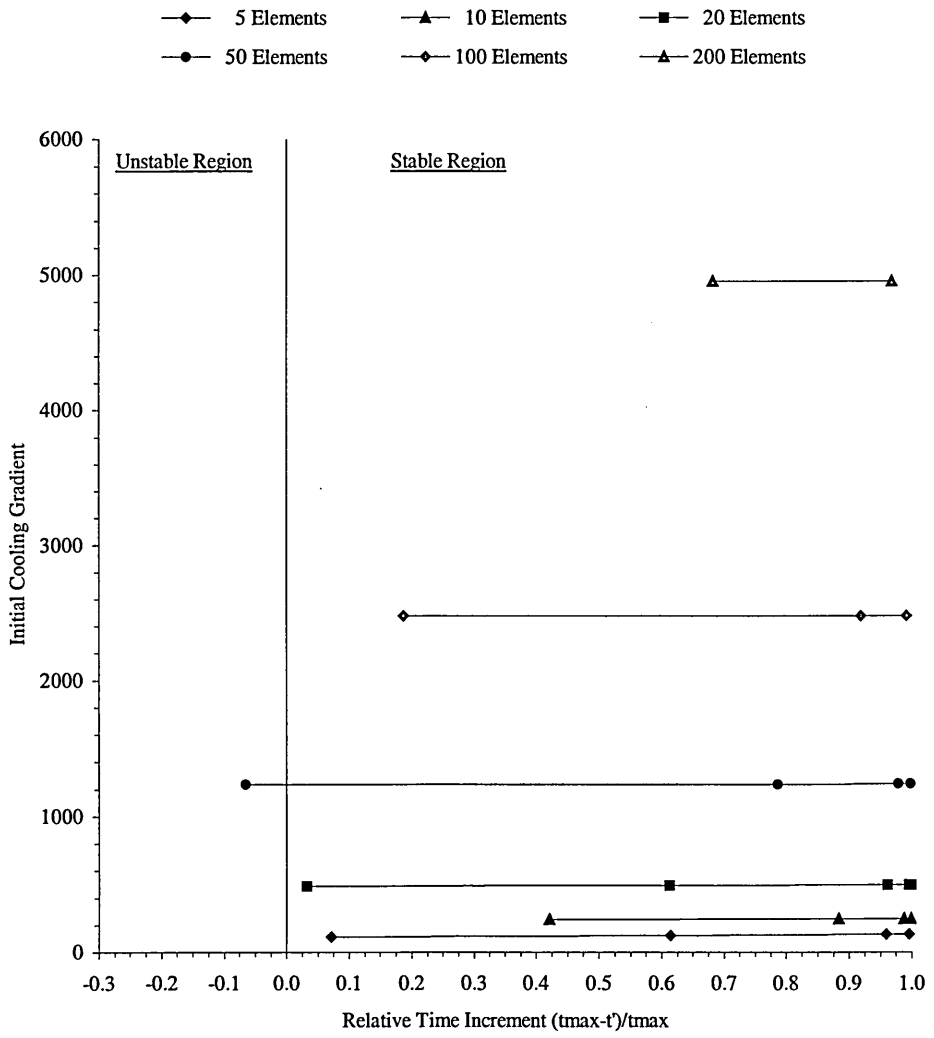


Figure 45: Relative time increment to initial cooling rate (dT/dt at t = 0s) for 650-350°C in 3s.

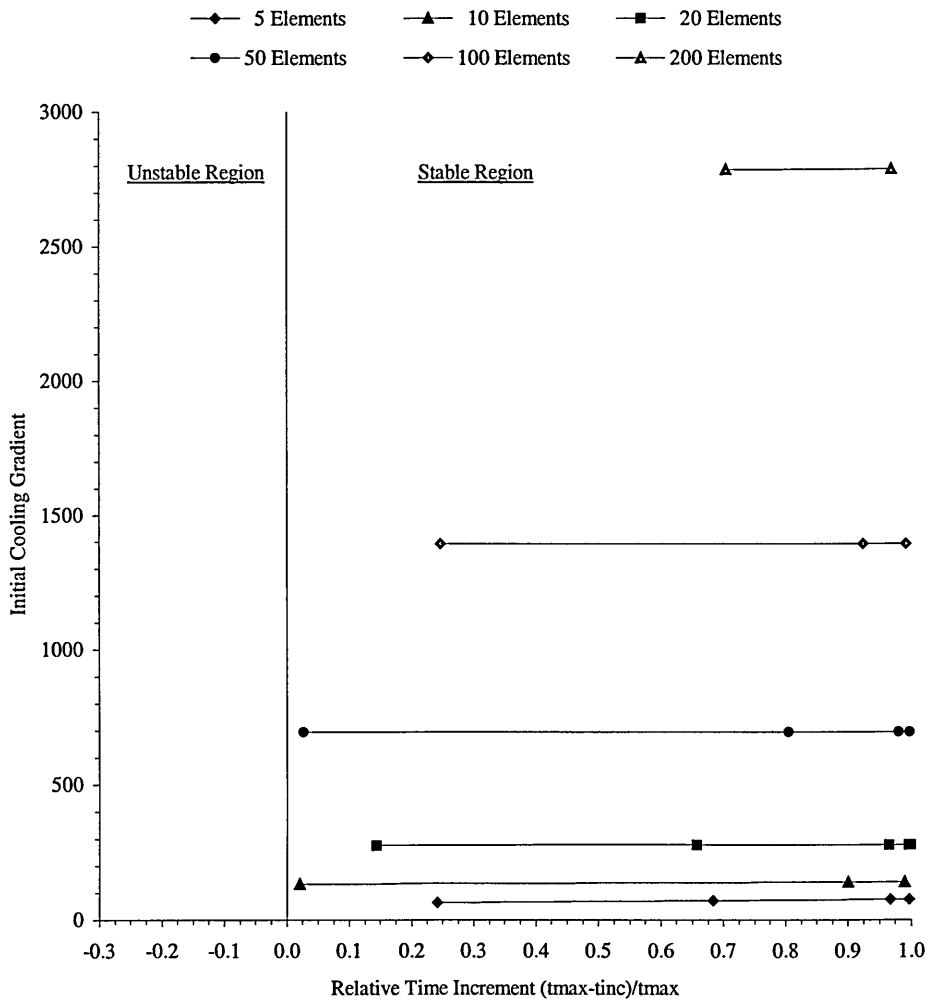


Figure 46: Relative time increment to initial cooling rate (dT/dt at t = 0s) for 550-350°C in 3s.

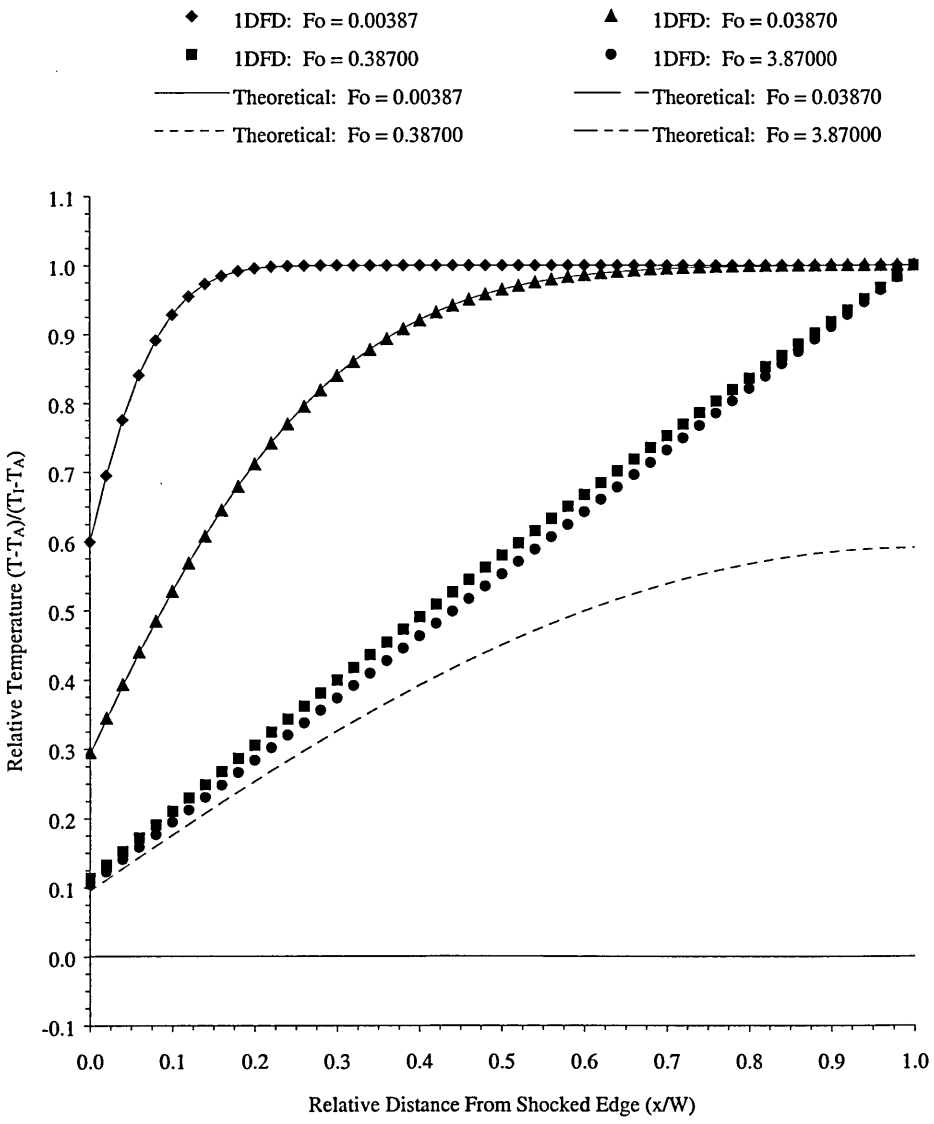


Figure 47: Finite difference model with fixed temperature insulated boundary condition to theoretical solution over small and large Fourier numbers.

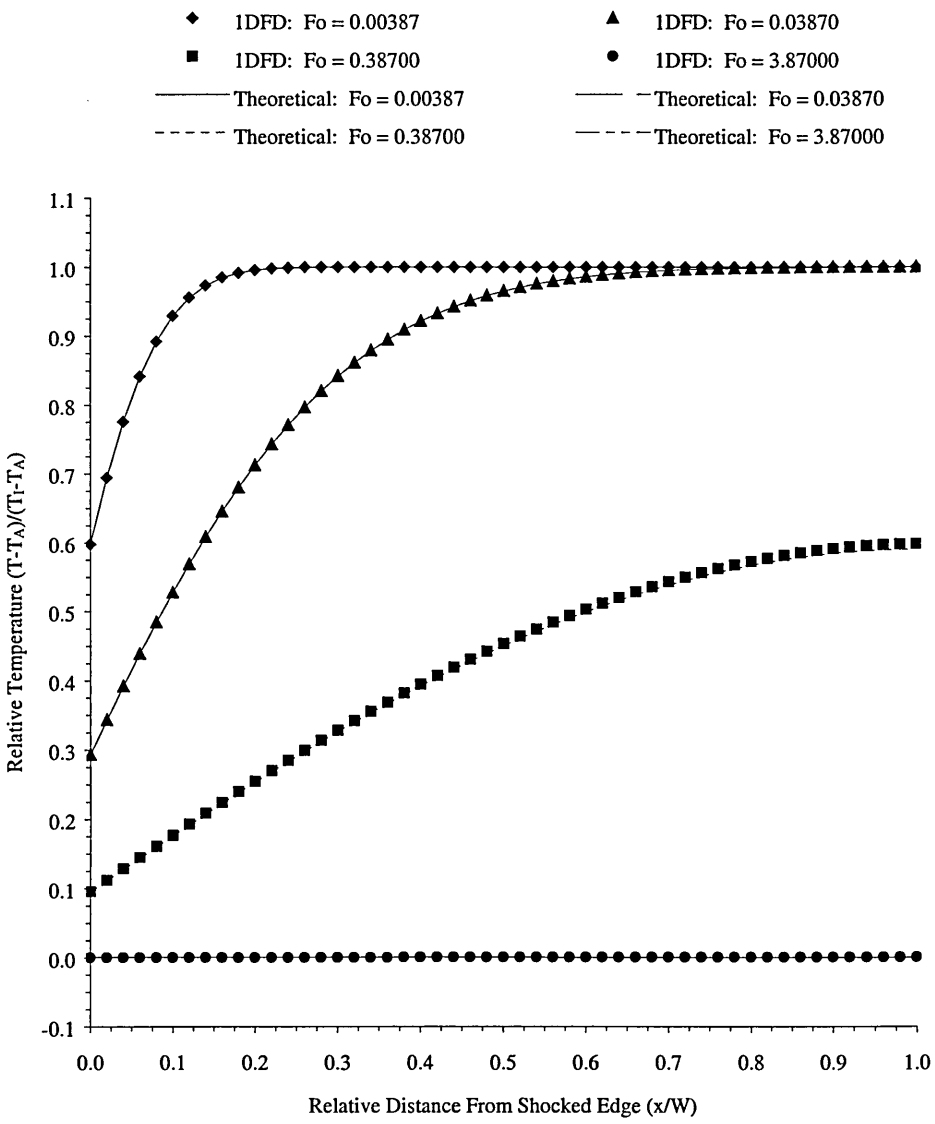


Figure 48: Finite difference model with dependant end temperature to theoretical solution over small and large Fourier numbers.

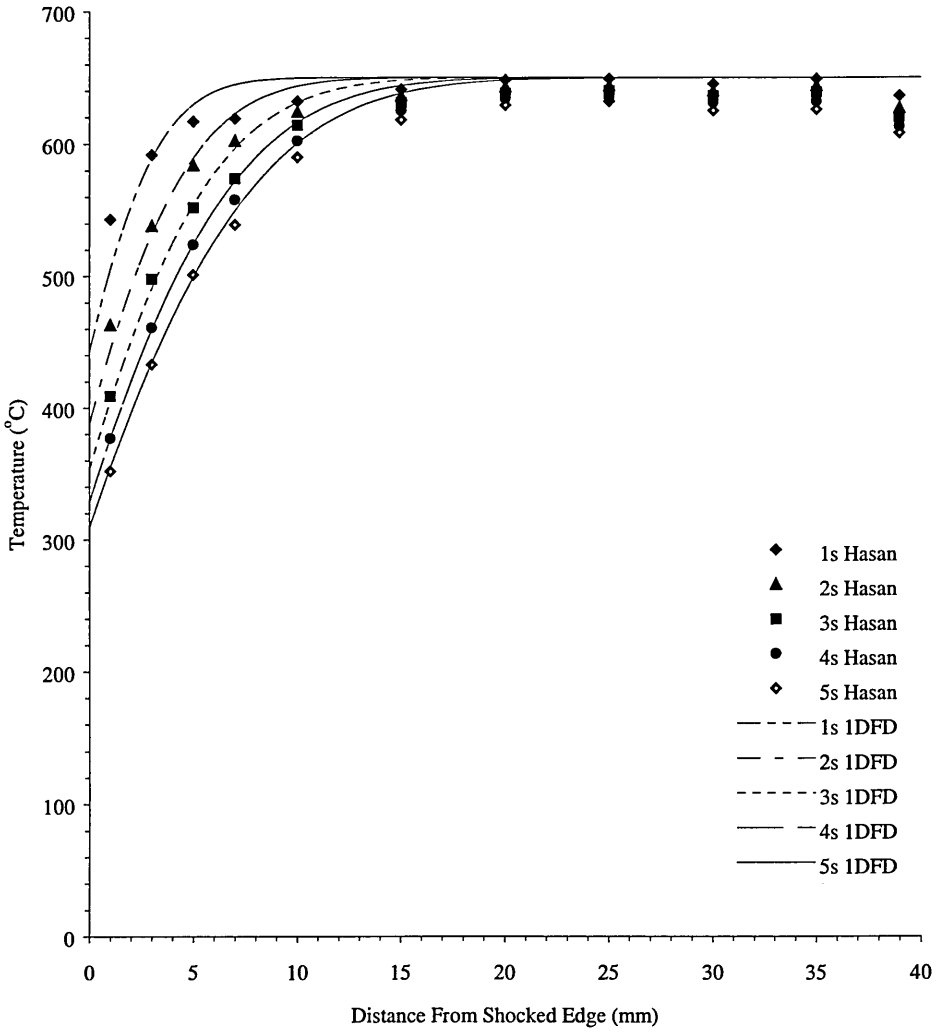


Figure 49: Single edge shock for Hasan and numerical at 650-310°C in 5s [124].

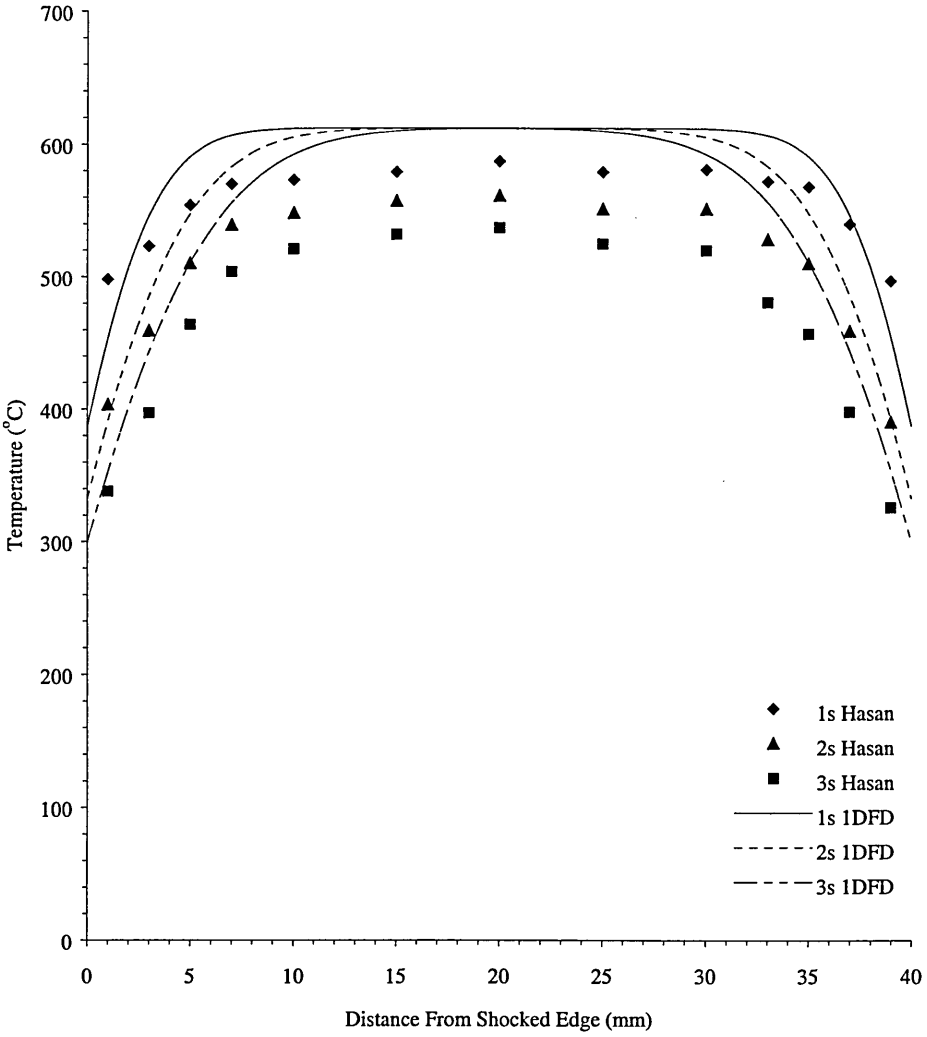


Figure 50: Double edge shock for Hasan and numerical at 612-300°C in 3s [124].

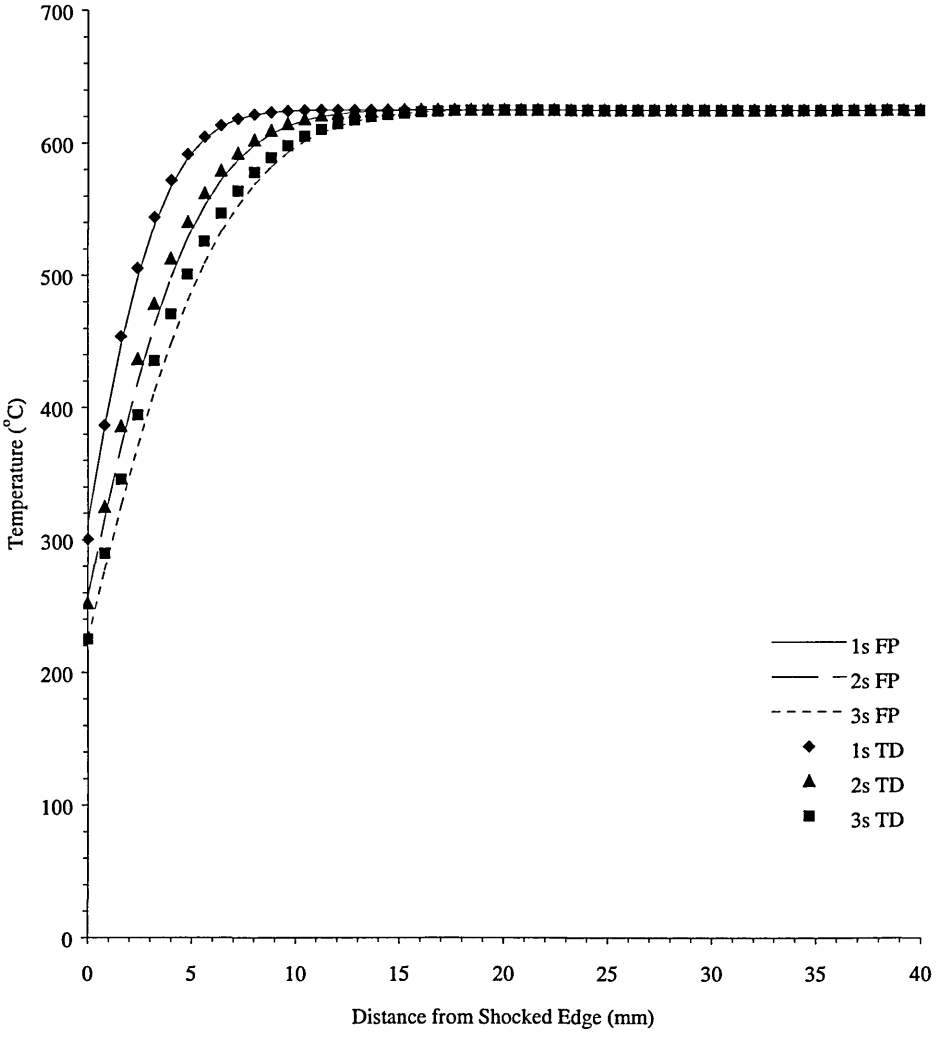


Figure 51: Effect of thermal property dependence on temperature for 625-225°C in 3s.

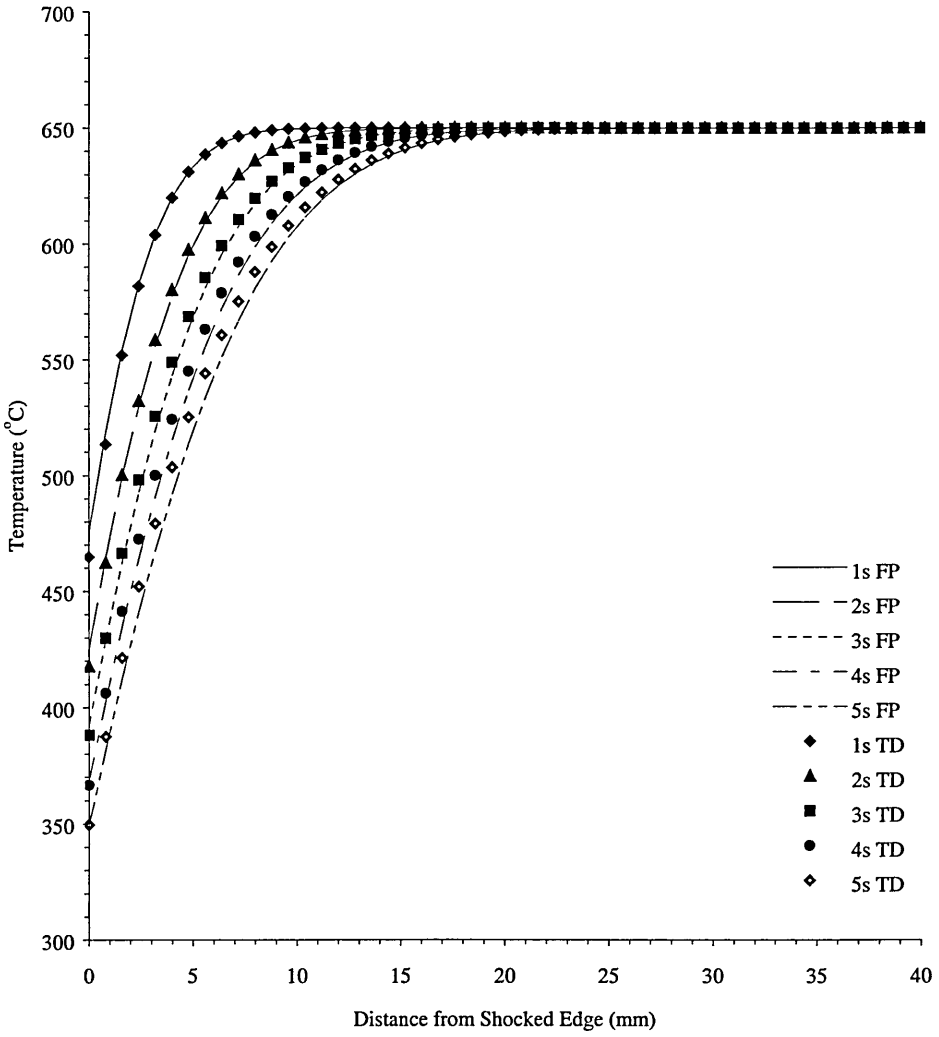


Figure 52: Effect of thermal property dependence on temperature for 650-350°C in 5s.

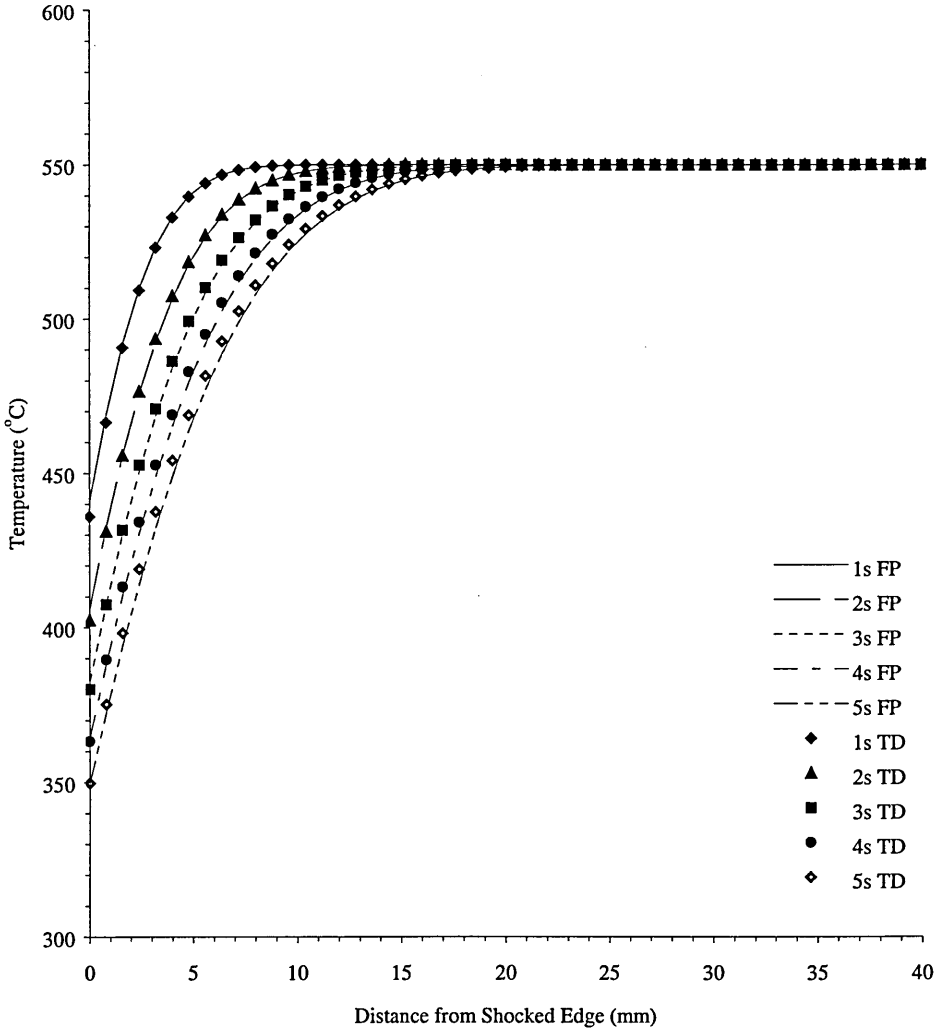


Figure 53: Effect of thermal property dependence on temperature for 550-350°C in 5s.

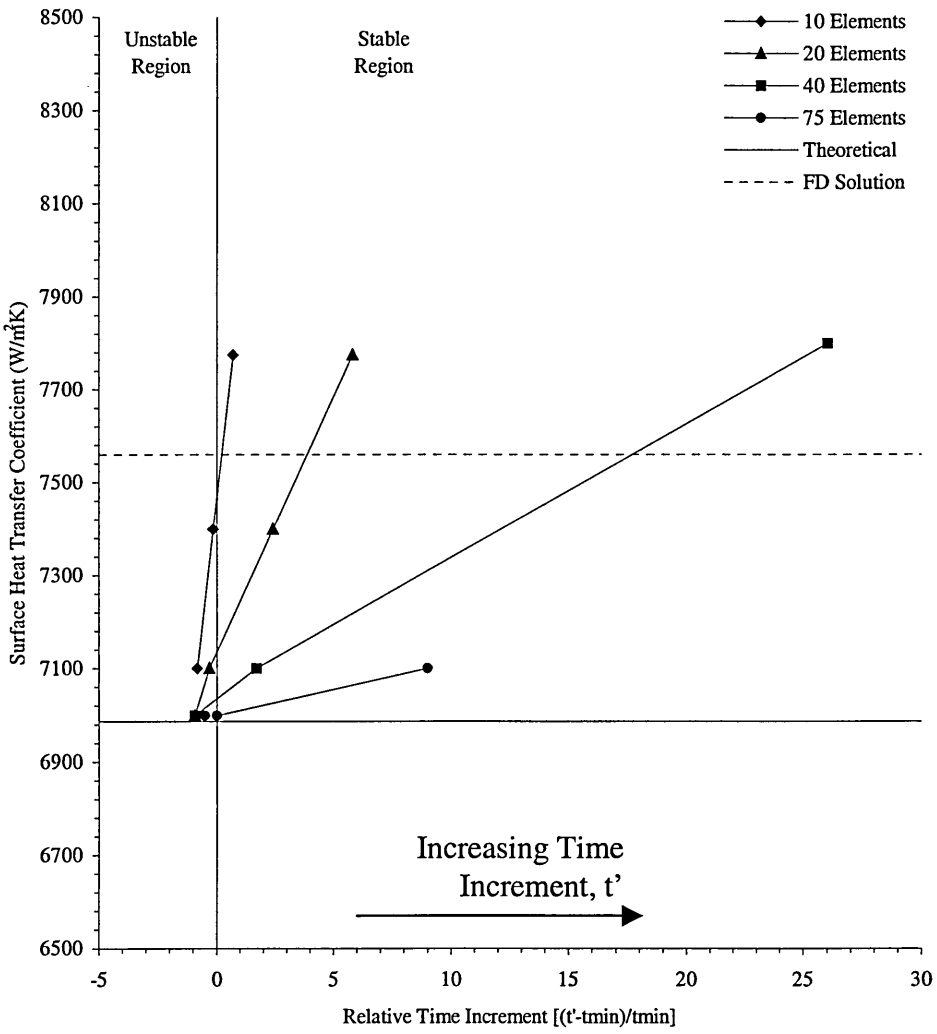


Figure 54: Space-time refinement of finite element model for 625-225°C in 3s.

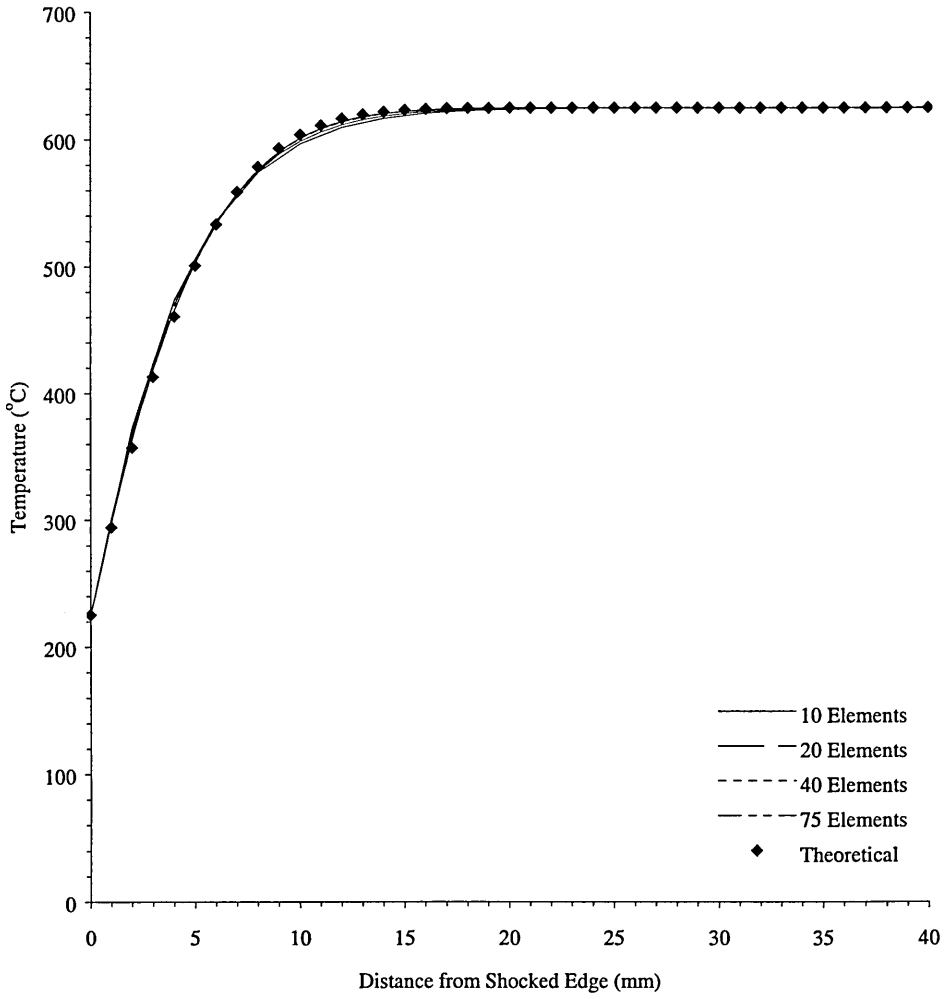


Figure 55: Thermal distribution refinement of finite element model for 625-225°C in 3s.

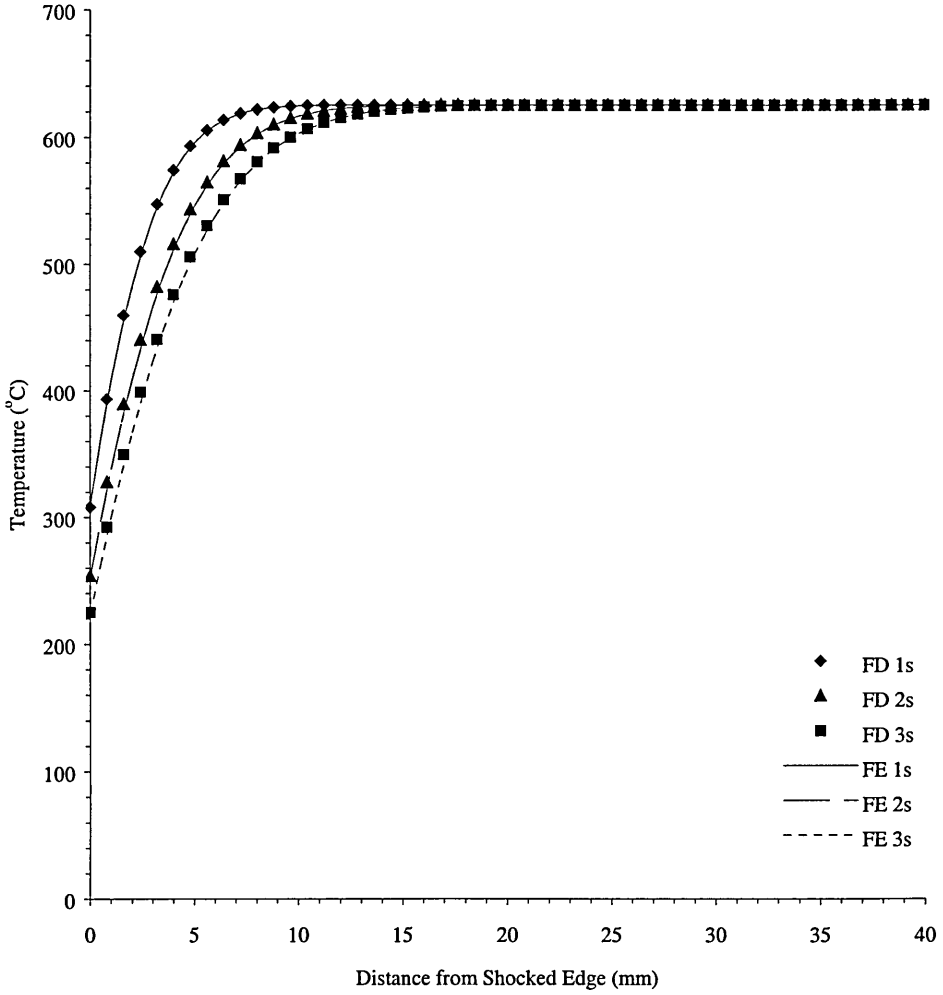


Figure 56: Comparison of space-time thermal distribution from finite difference & finite element models for 625-225°C in 3s.

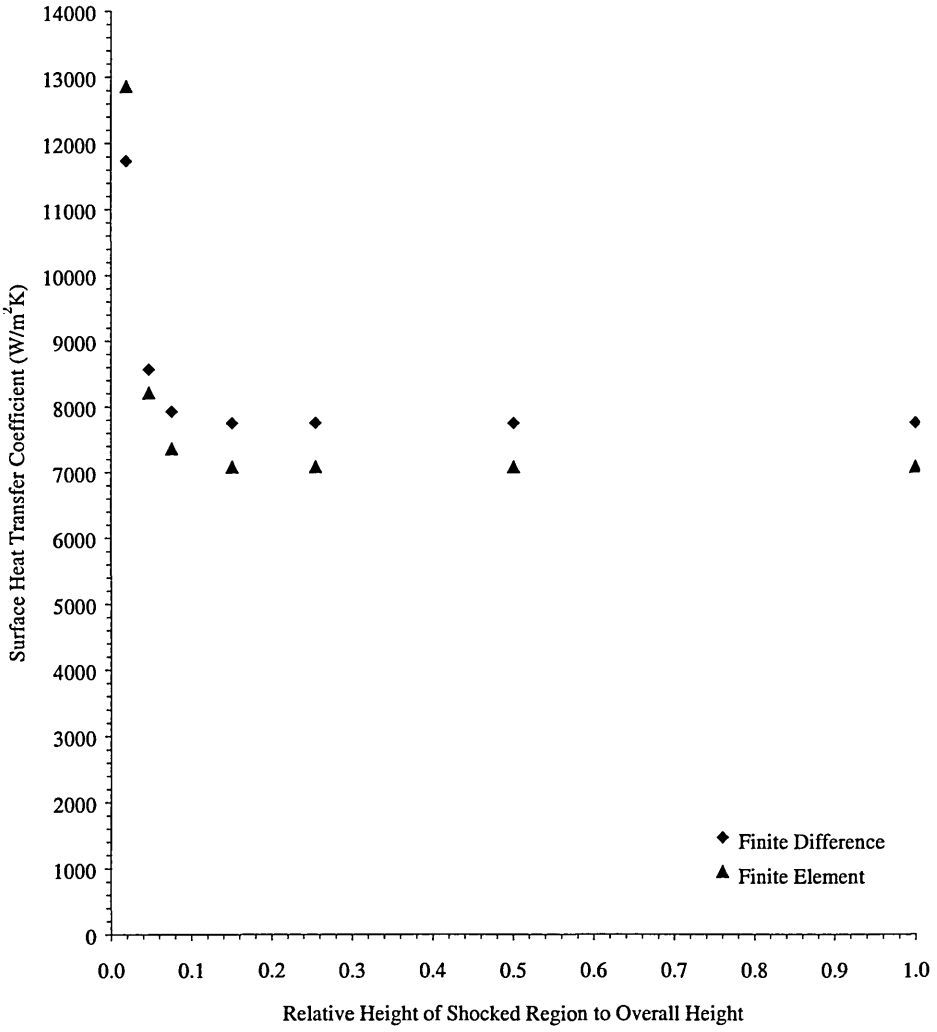


Figure 57: Effect of shock localisation on surface heat transfer coefficient.

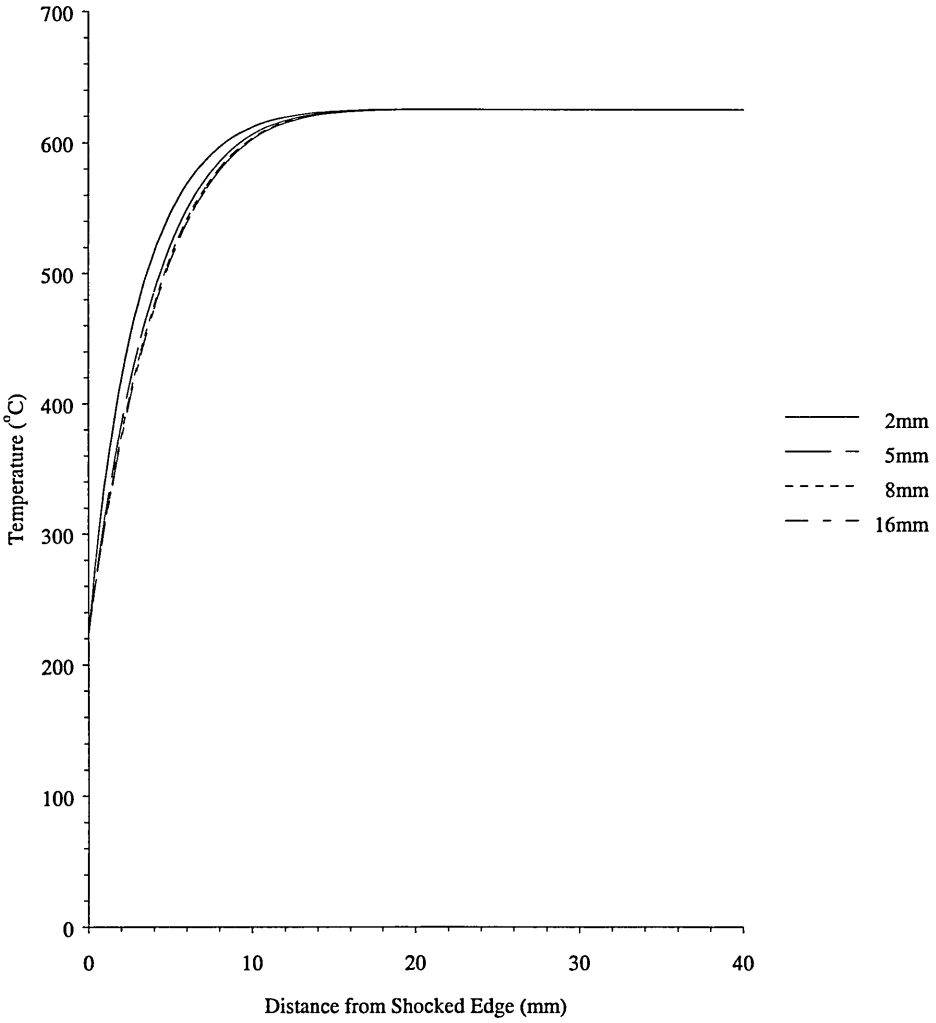


Figure 58: Effect of localisation on thermal distribution for finite difference for 625-225°C 3s.

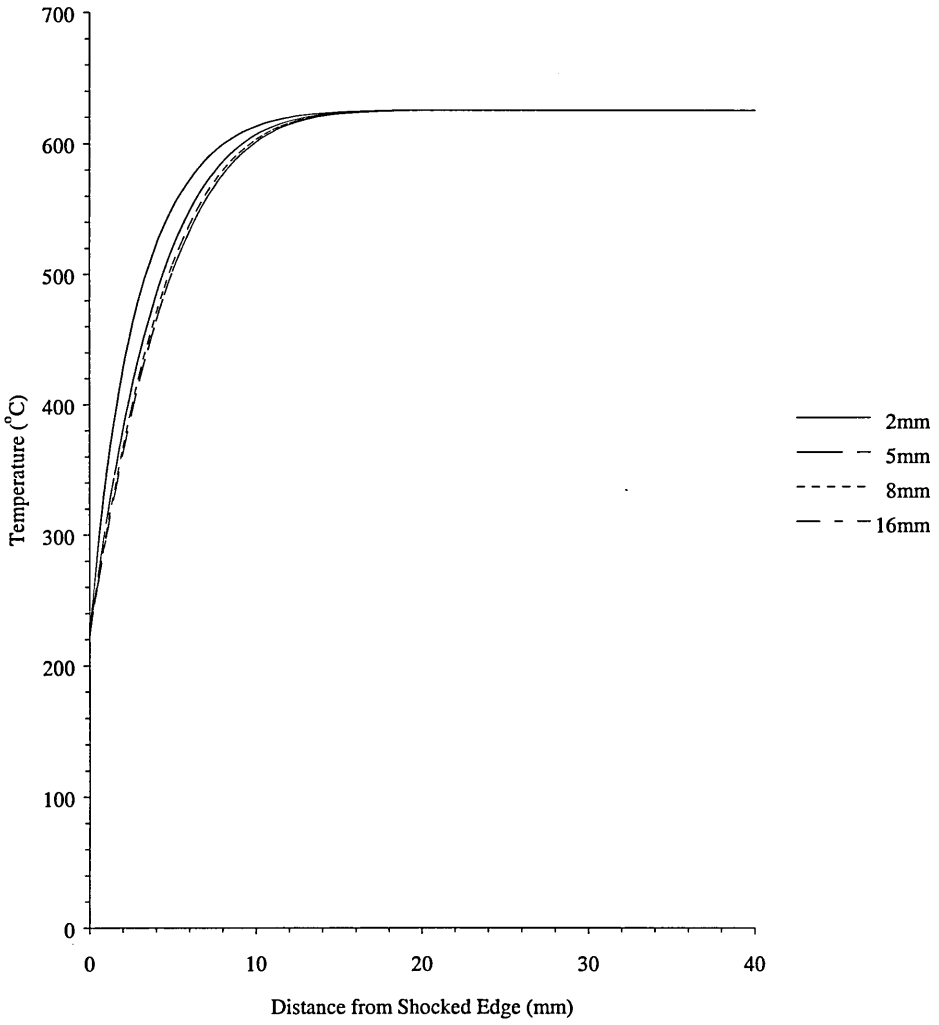


Figure 59: Effect of localisation on thermal distribution for finite element model for 625-225°C in 3s.

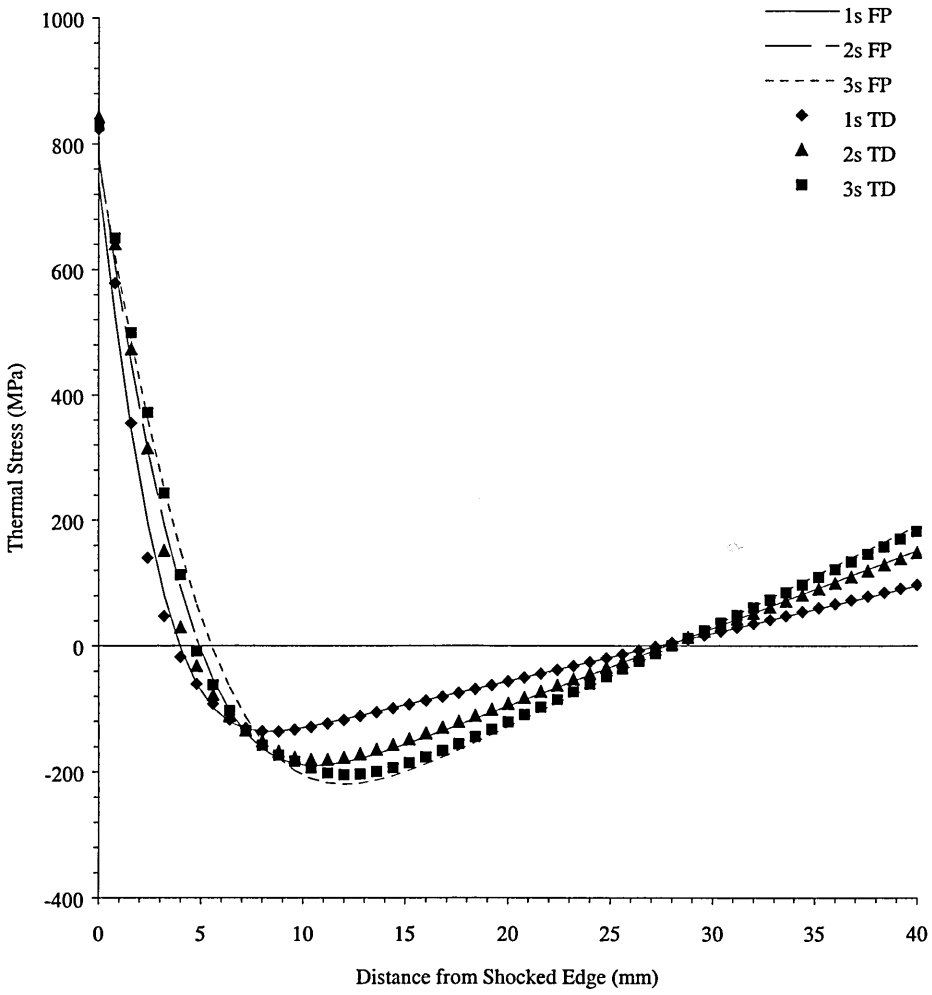


Figure 60: One dimensional elastic stresses by Timoshenko for fixed (FP) & temperature dependent (TD) thermomechanical properties at 625-225°C 3s.

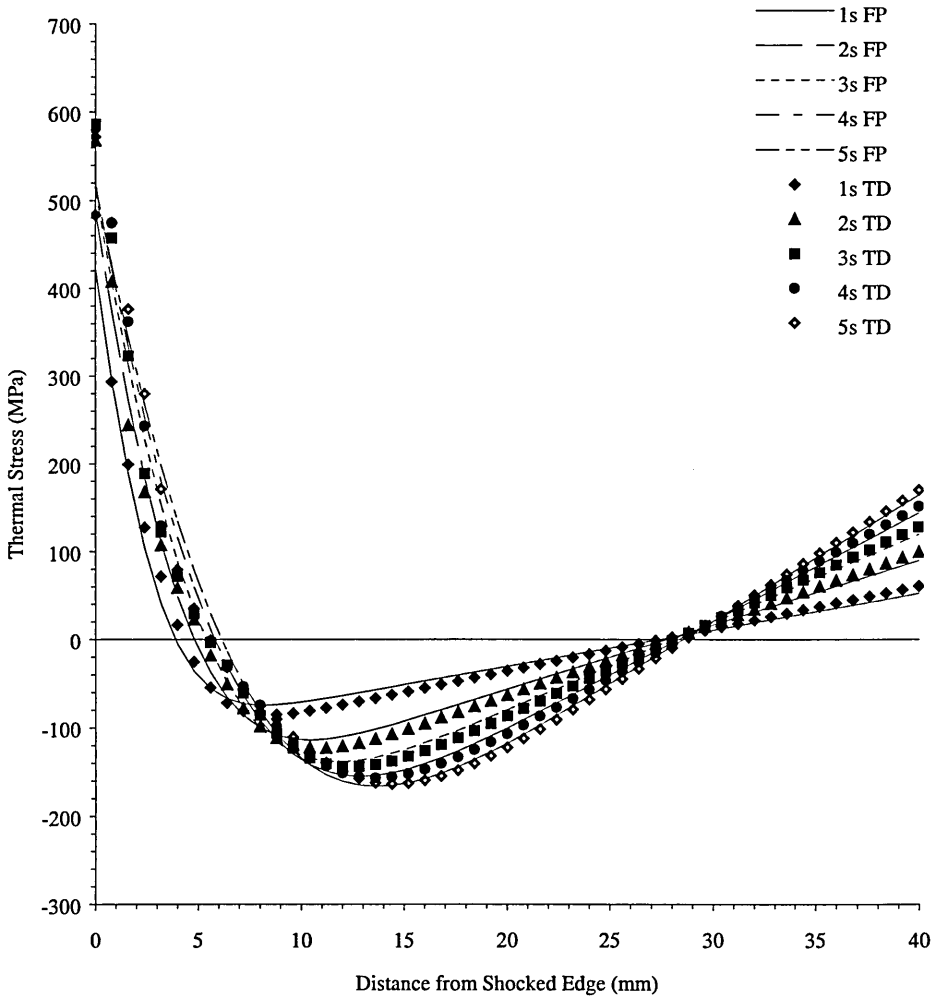


Figure 61: One dimensional elastic stresses by Timoshenko for fixed (FP) & temperature dependent (TD) thermomechanical properties at 650-350°C 5s.

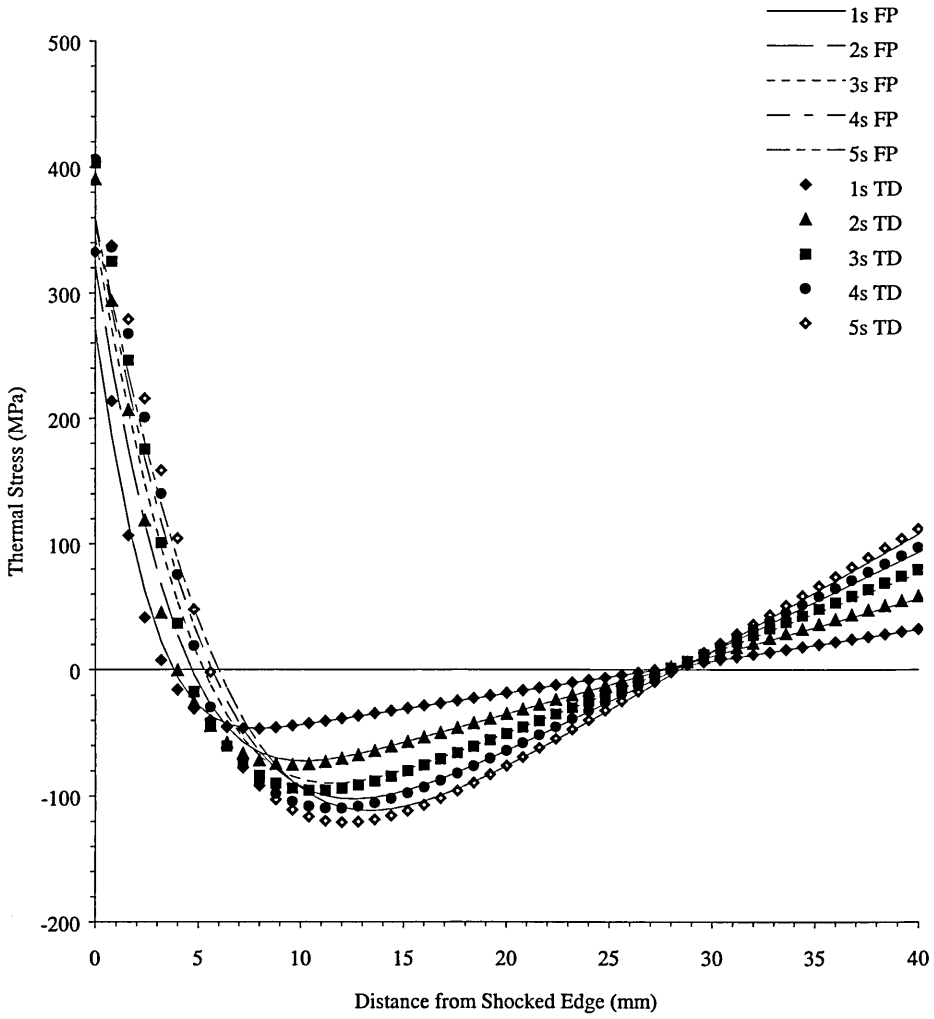


Figure 62: One dimensional elastic stresses by Timoshenko for fixed (FP) & temperature dependent (TD) thermal properties at 550-350°C 5s.

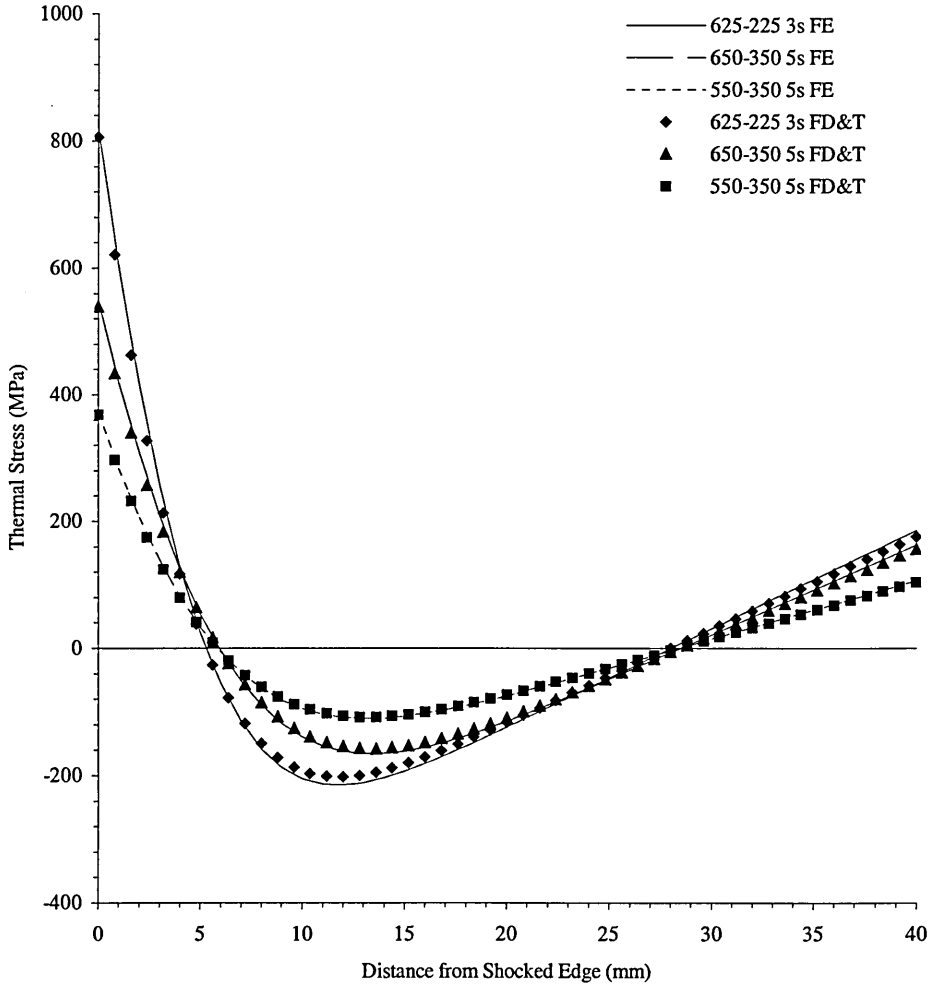


Figure 63: End-of-Cycle stress profiles for fixed elastic properties for 2DFD/Timoshenko and 2DFE models.

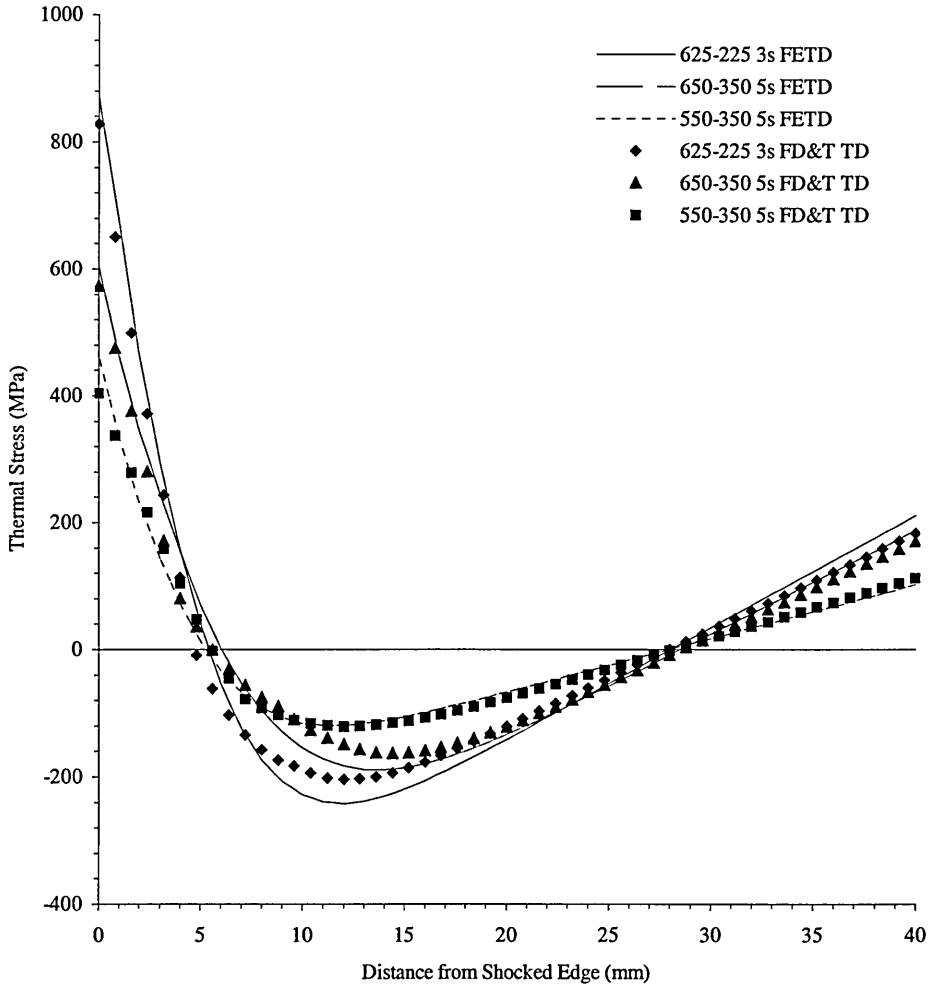


Figure 64: End-of-Cycle stress profiles for temperature dependant elastic properties for 2DFD/Timoshenko and 2DFE models.

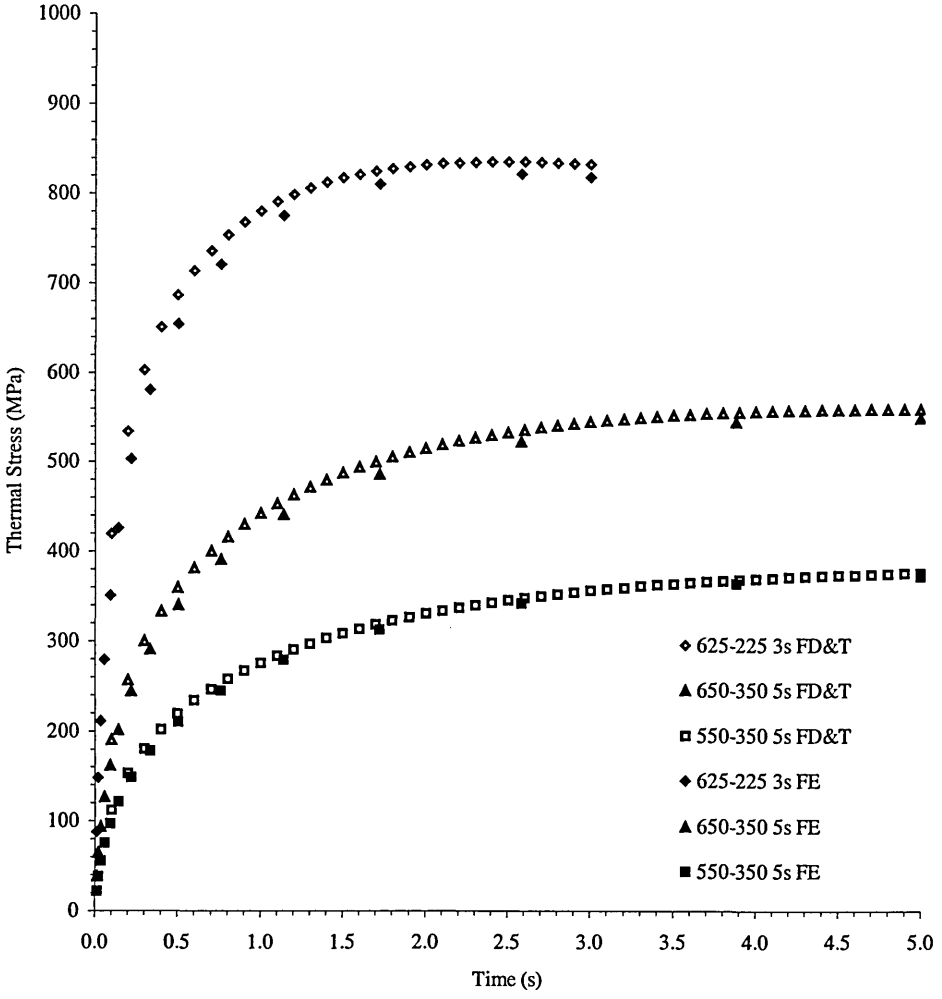


Figure 65: Change in elastic stress through time for full-edge downshock by temperature dependant thermal distribution & fixed mechanical property solution.

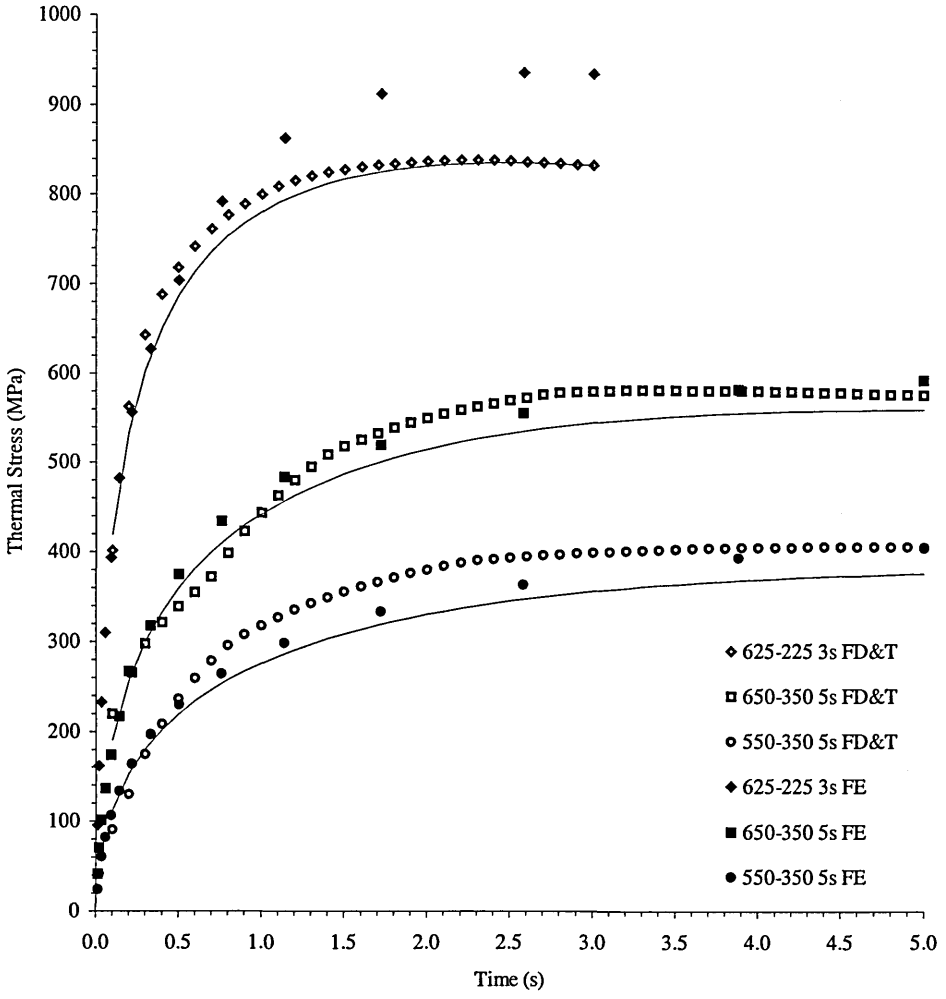


Figure 66: Change in elastic stress through time for full-edge downshock by temperature dependant Timoshenko/2DFD thermal distribution & temperature dependant finite element solution.
Note: solid lines indicate FD solution of Figure 65.

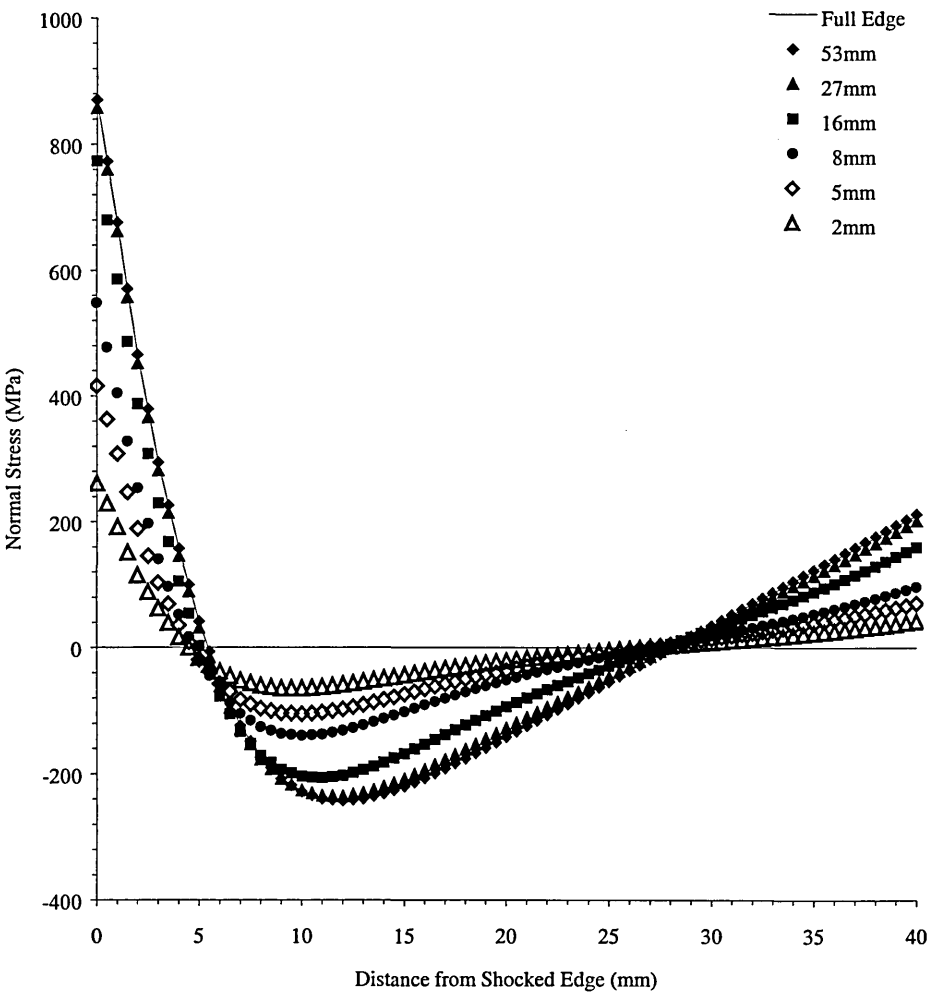


Figure 67: Effect of localisation on normal stress for 625-225°C in 3s by FE.

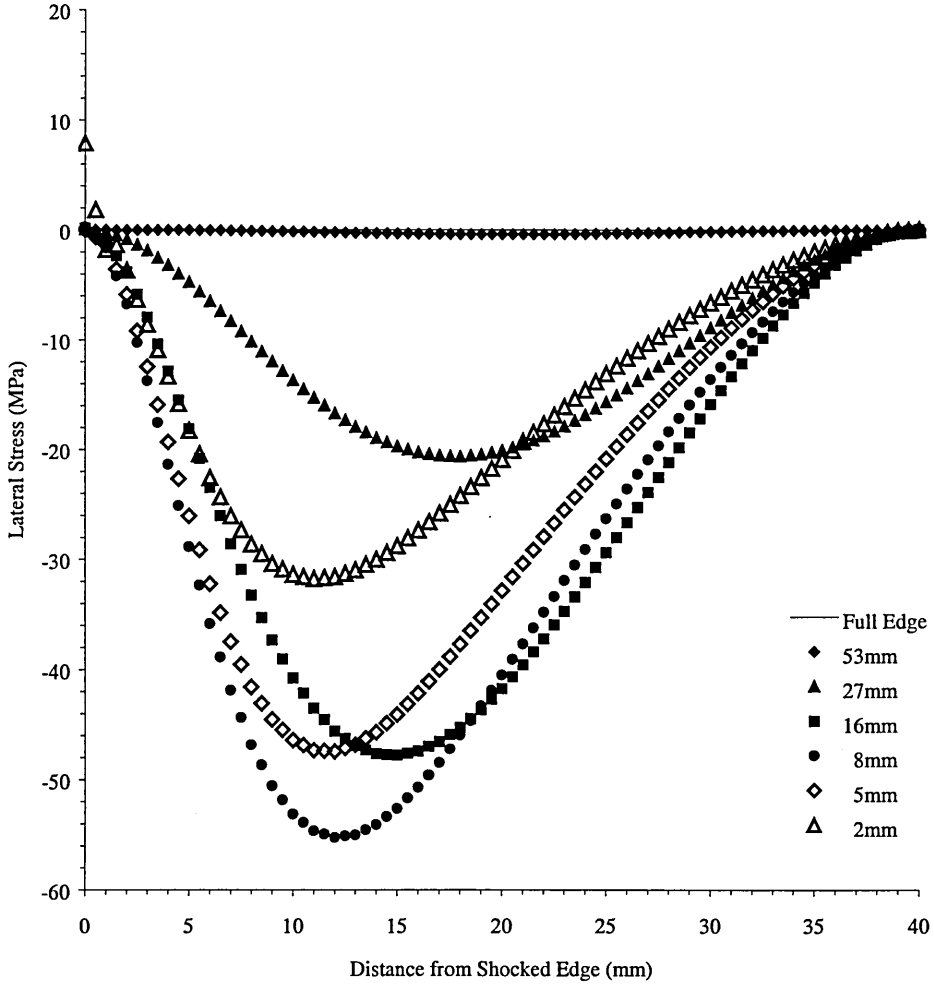


Figure 68: Effect of localisation on lateral stress for 625-225°C in 3s by FE.

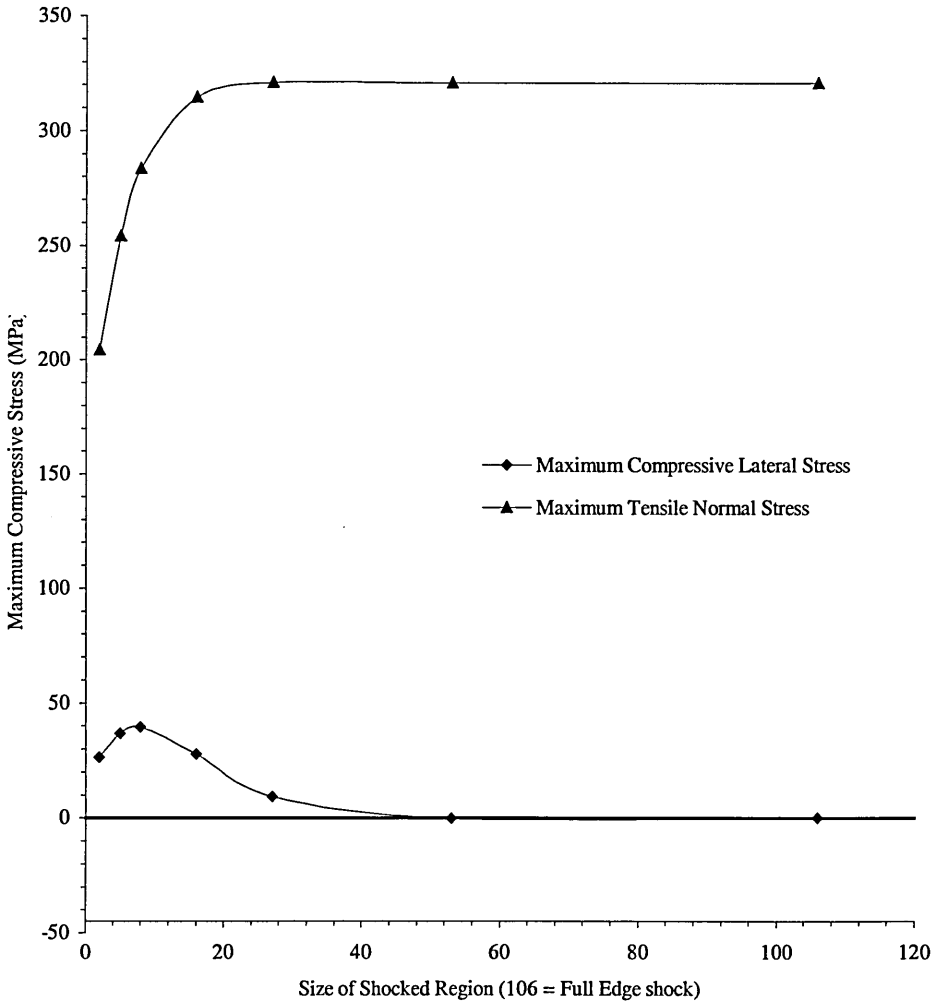


Figure 69: Change in maximum compressive stress through increasing localisation by FE.

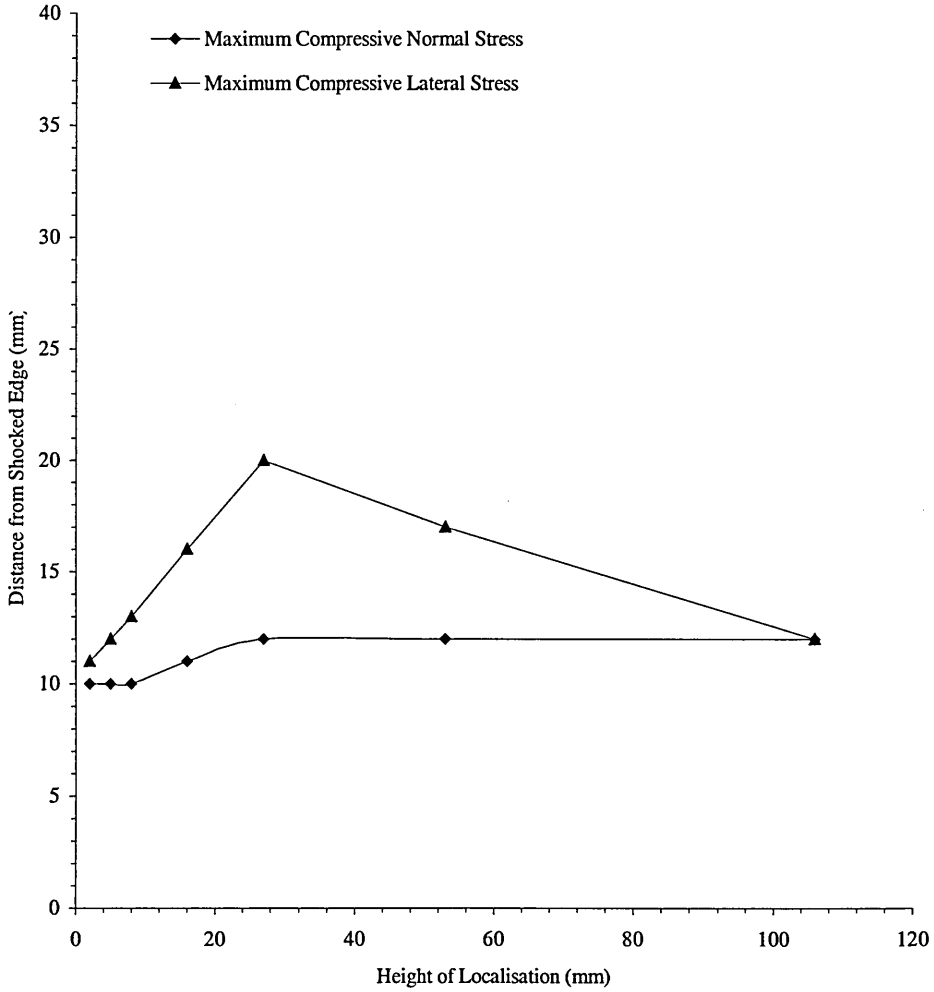


Figure 70: Position of normal & lateral maximum compressive stress for various localisations by FE.

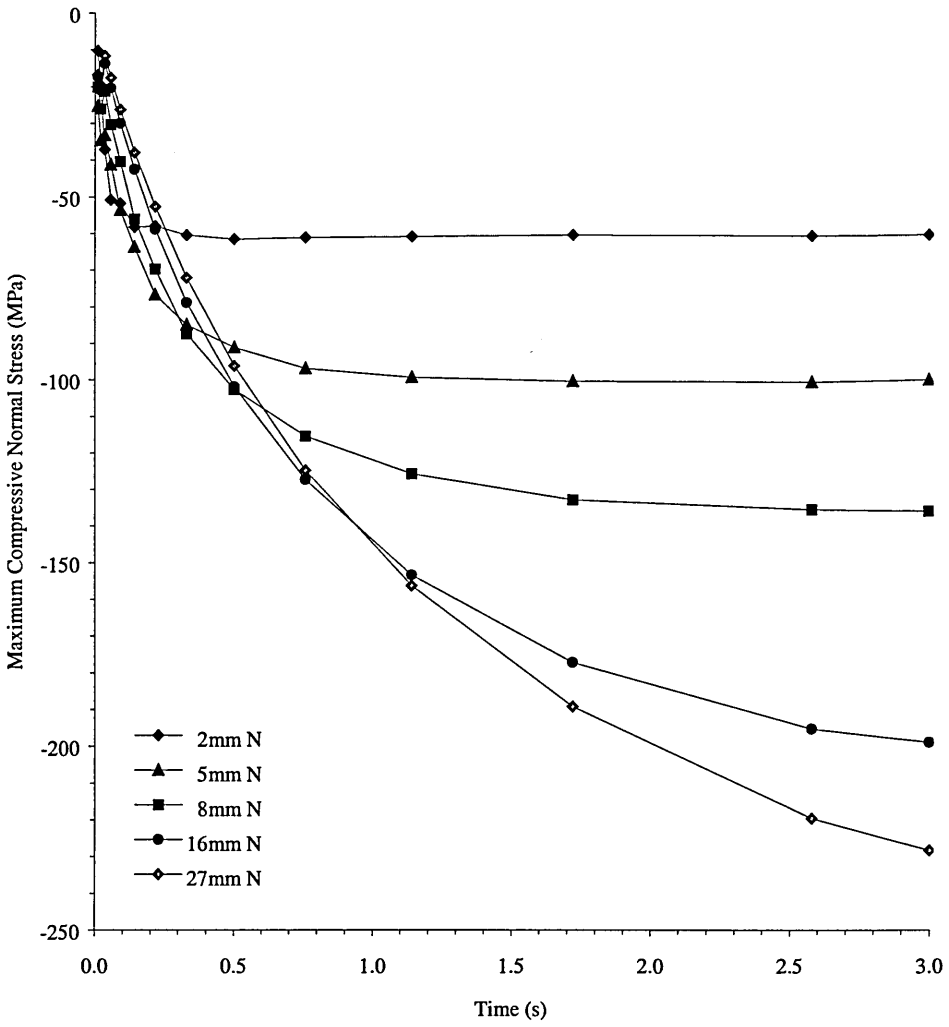


Figure 71: Change in maximum compressive normal stress through time for 625-225°C in 3s by FE.

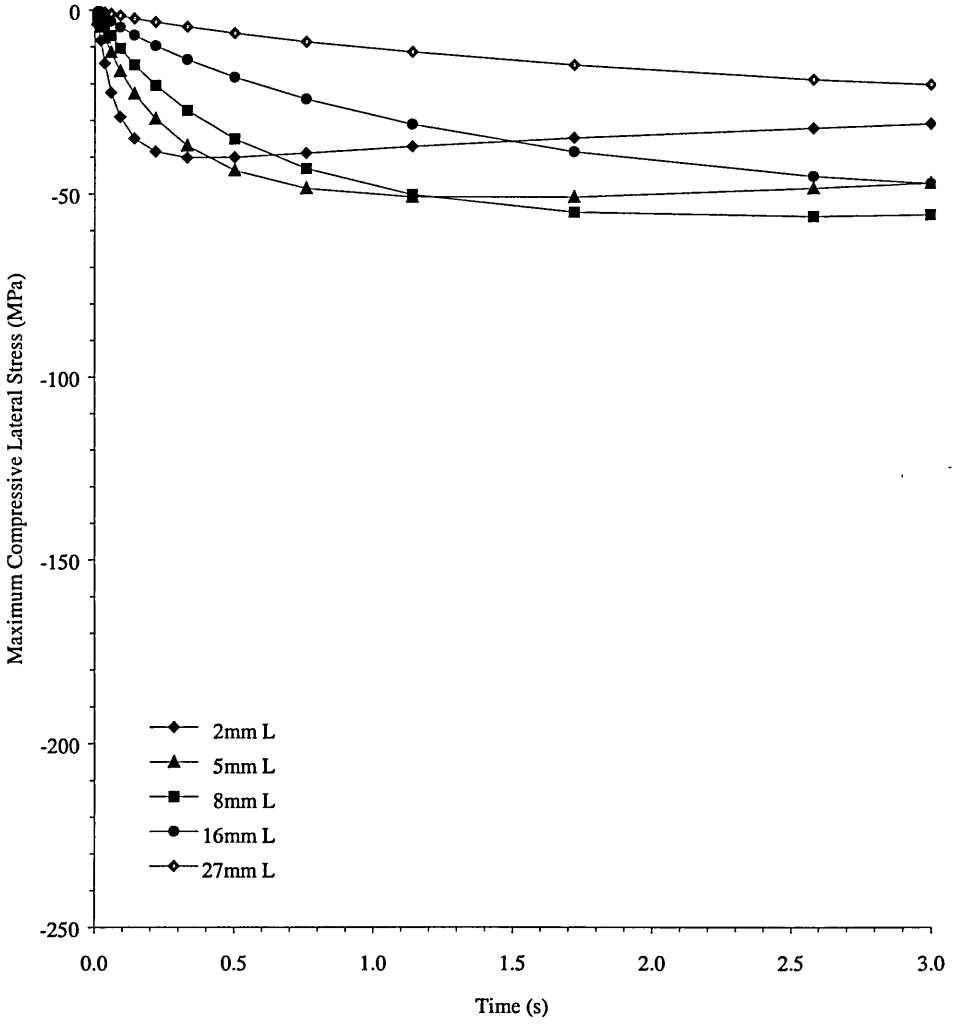


Figure 72: Change in maximum compressive lateral stress through time for 625-225°C in 3s by FE.

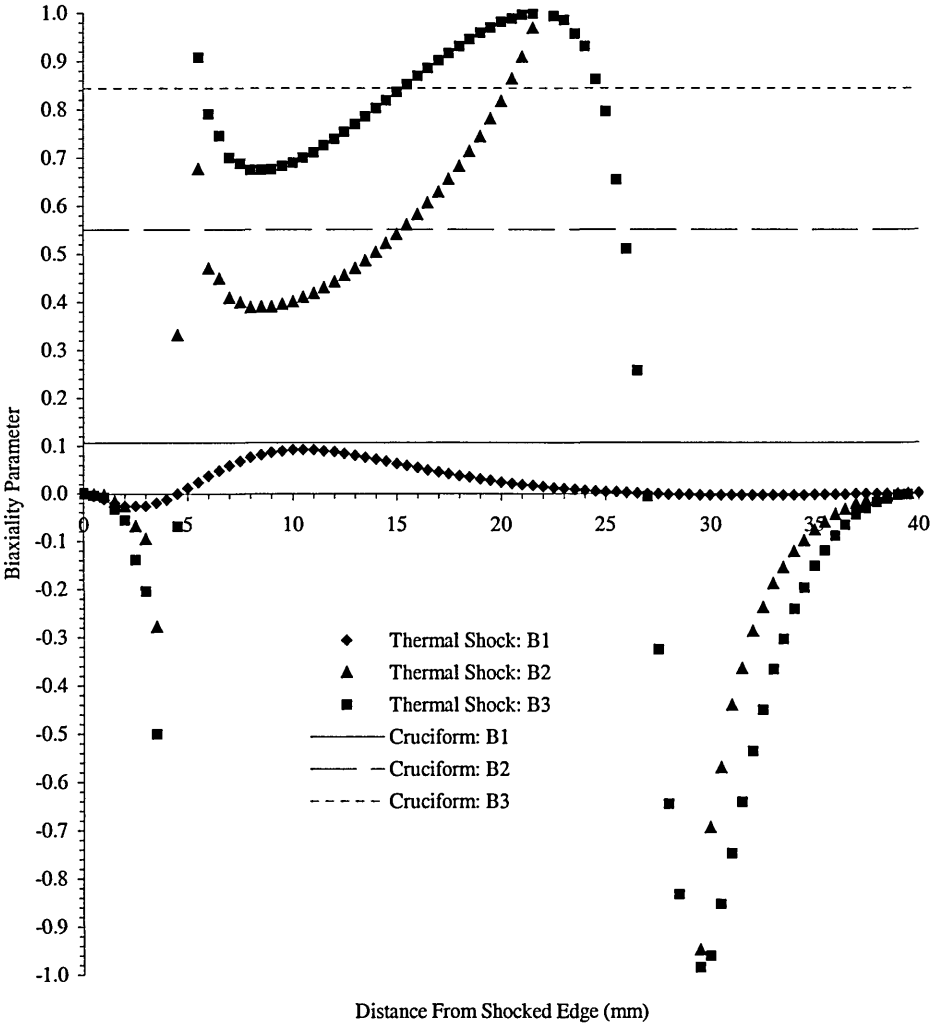


Figure 73: Biaxiality along symmetry plane for 625-225°C in 3s shock for end-of-cycle stresses by FE.

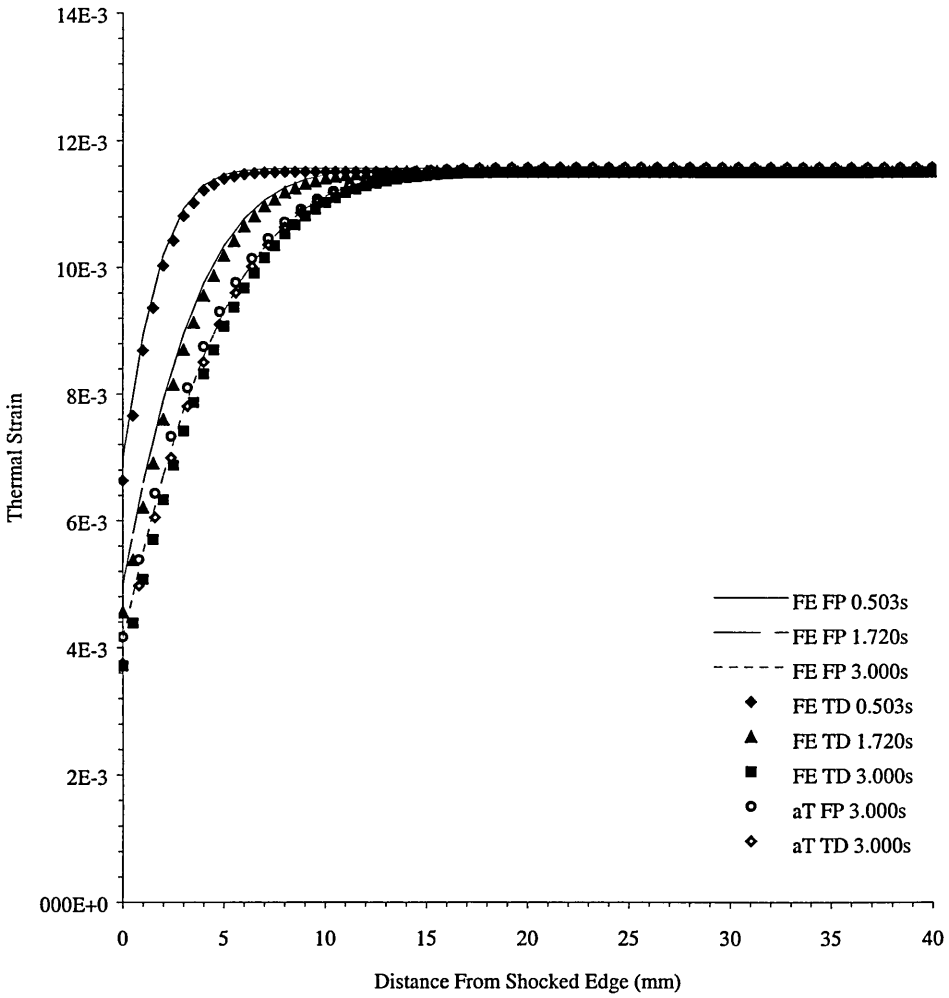


Figure 74: Thermal strains through time for 625-225°C in 3s
for fixed property and temperature dependent thermal expansion.

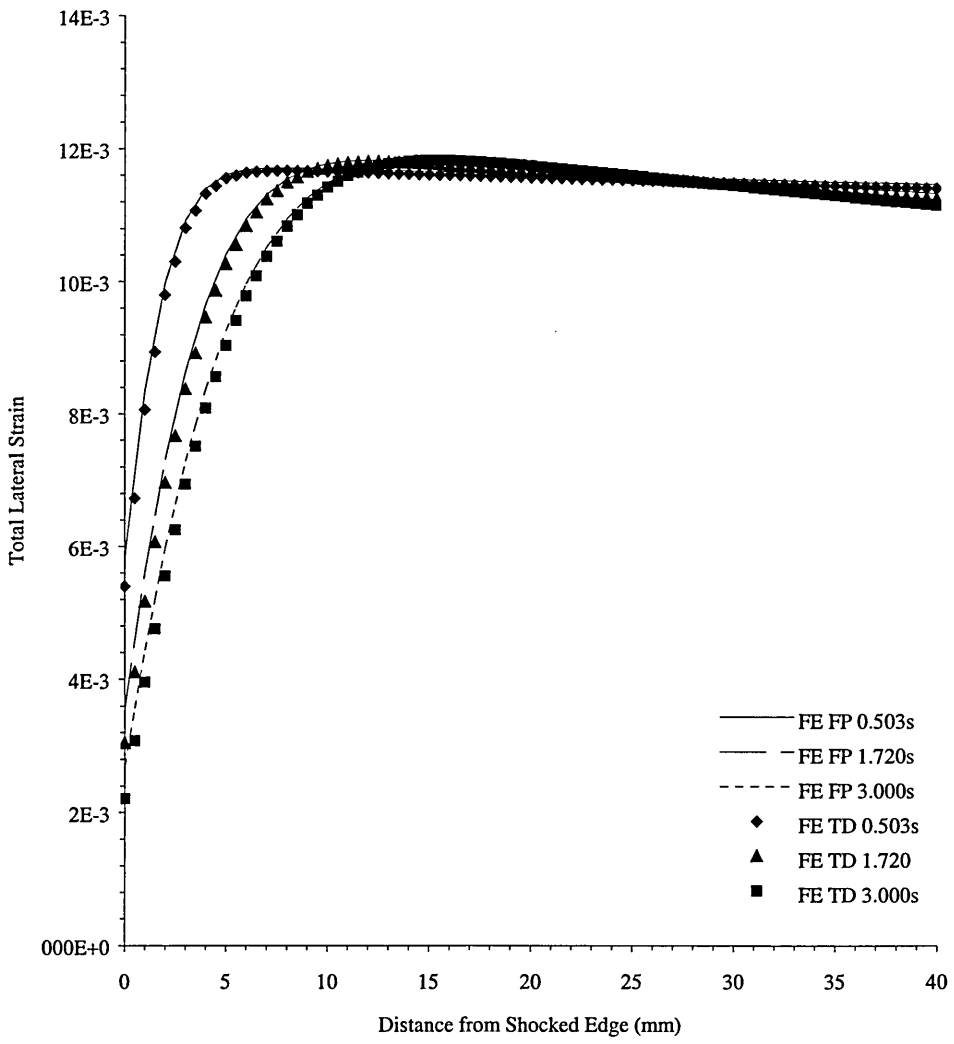


Figure 75: Total lateral strain through time for 625-225°C in 3s shock for fixed property and temperature dependent thermal expansion.

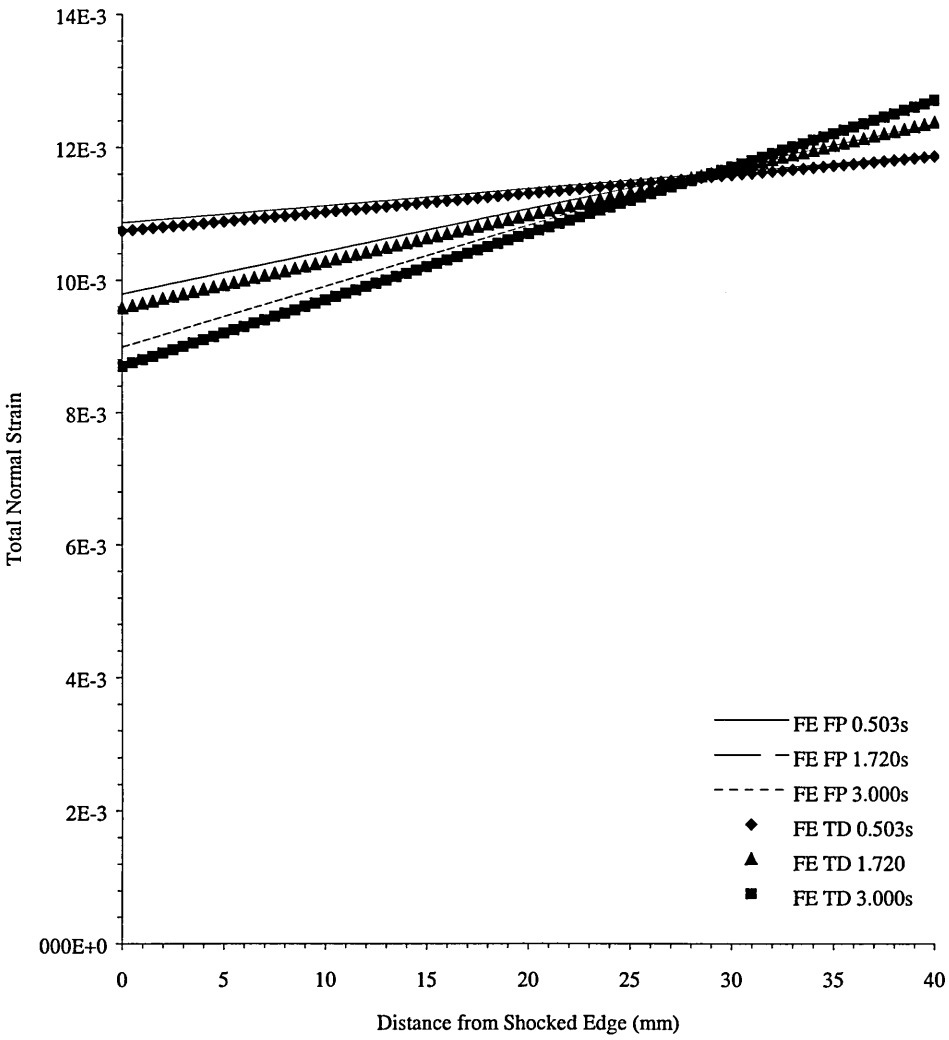


Figure 76: Total normal strain through time for 625-225°C in 3s shock for fixed property and temperature dependent thermal expansion.

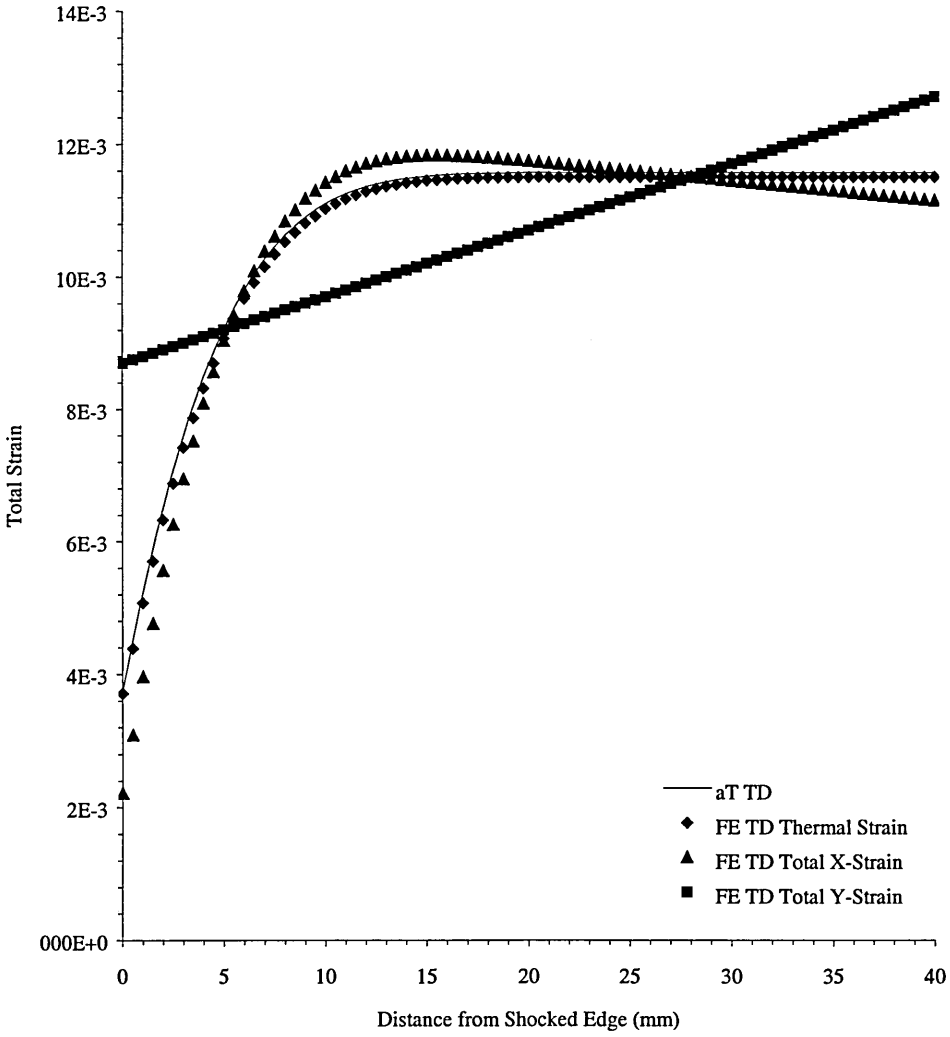


Figure 77: Total strains for 625-225°C in 3s shock
for temperature dependent thermal expansion.

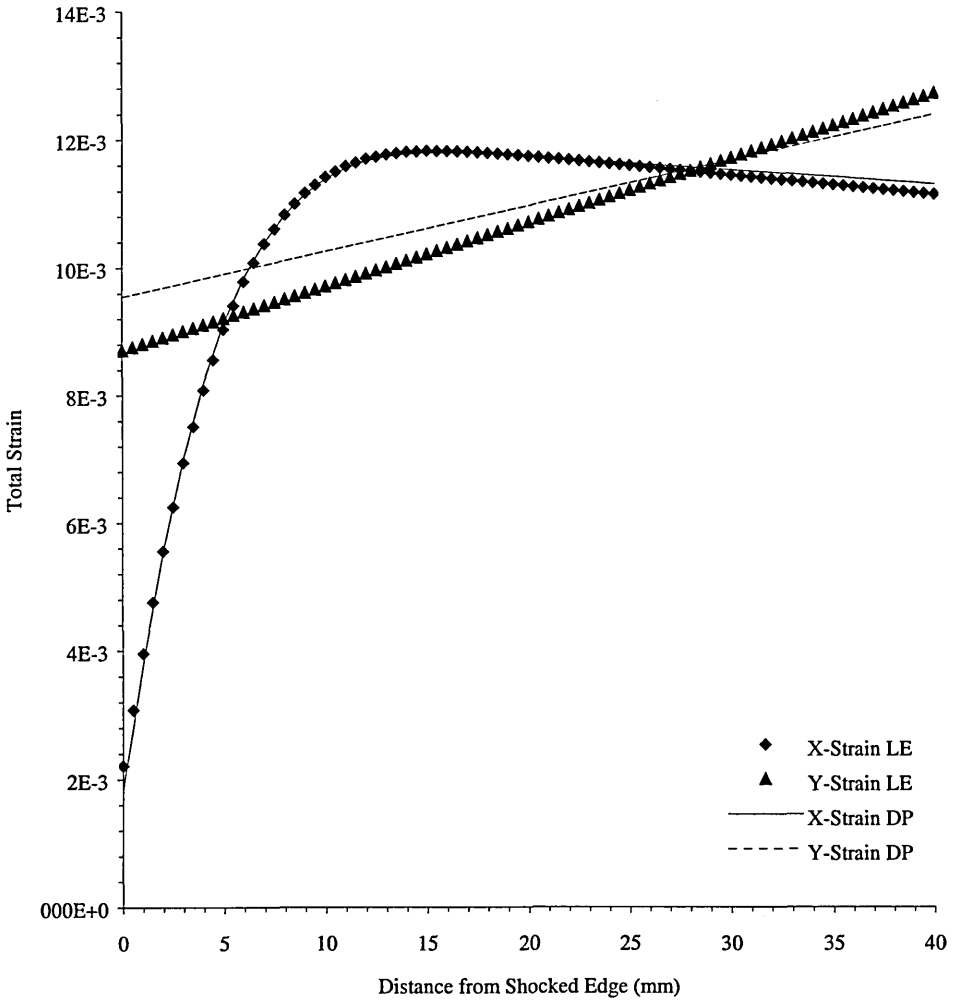


Figure 78: Effect of plasticity on total strain for 625-225°C in 3s by FE.

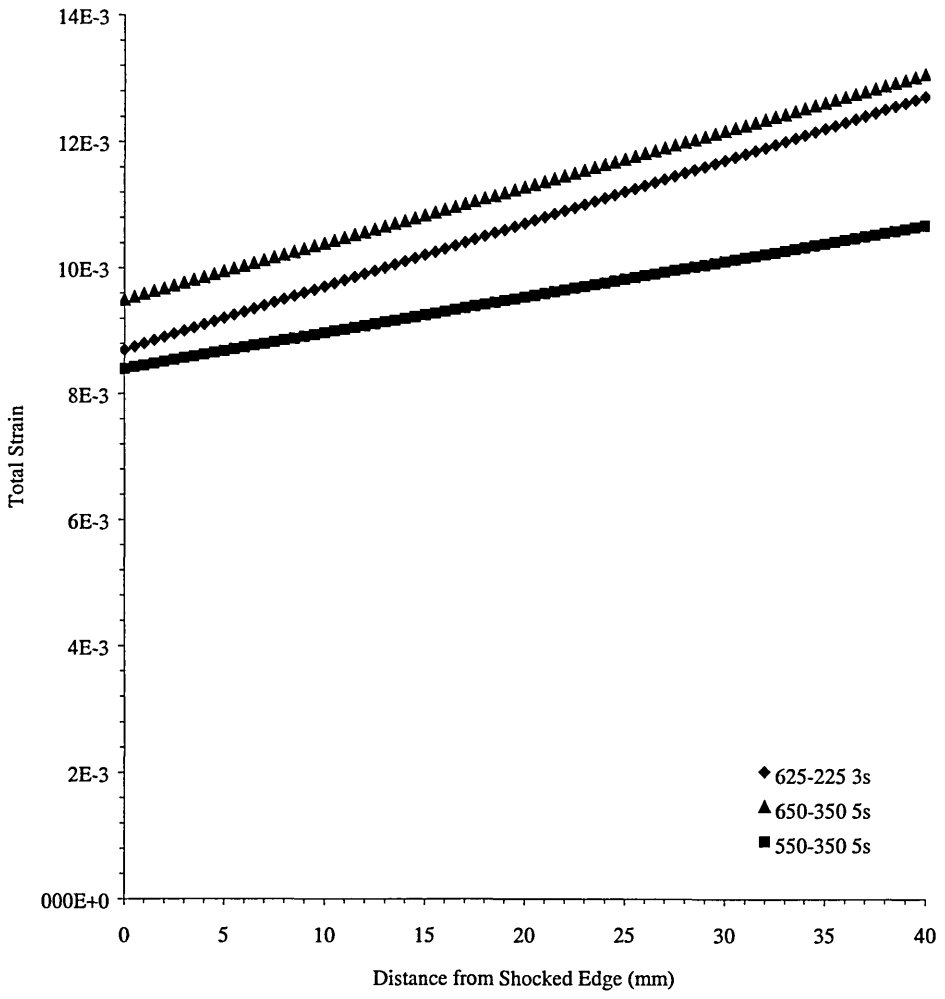


Figure 79: Normal strain at three separate levels of shock by FE.

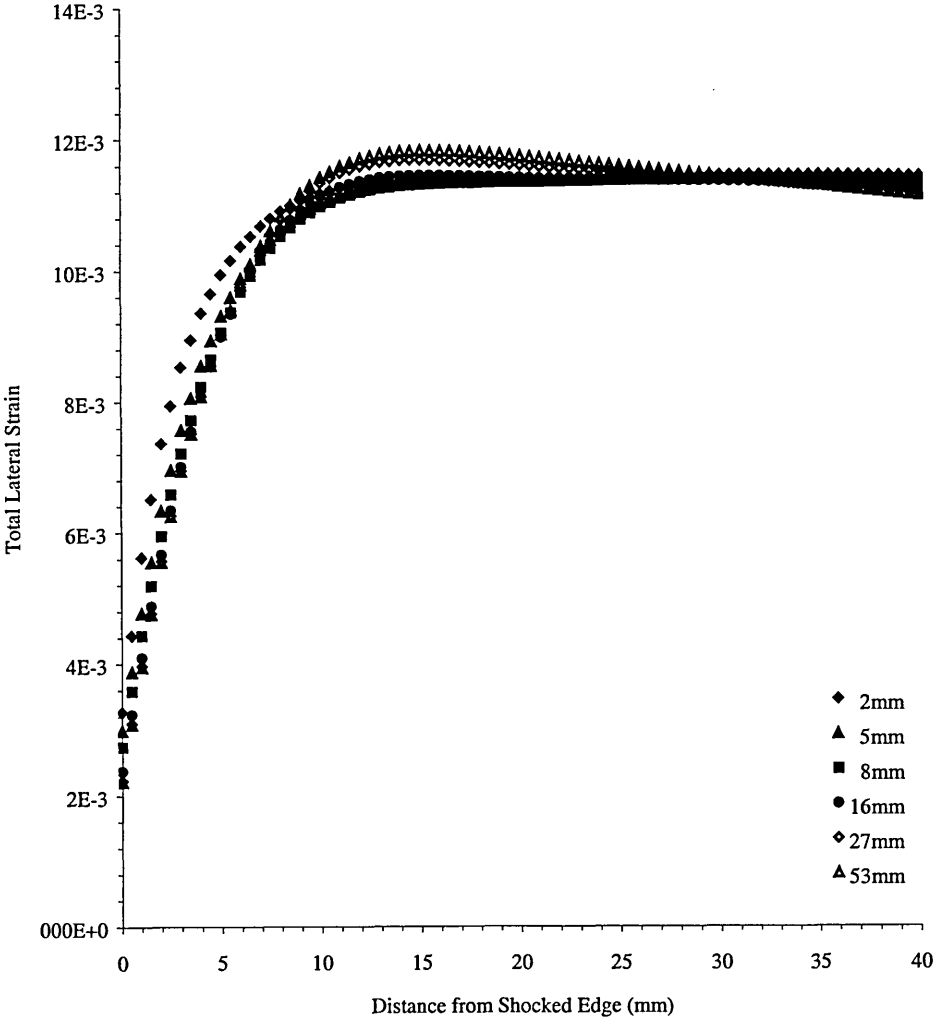


Figure 80: Effect of localisation on lateral strains for 625-225°C 3s shock by FE.

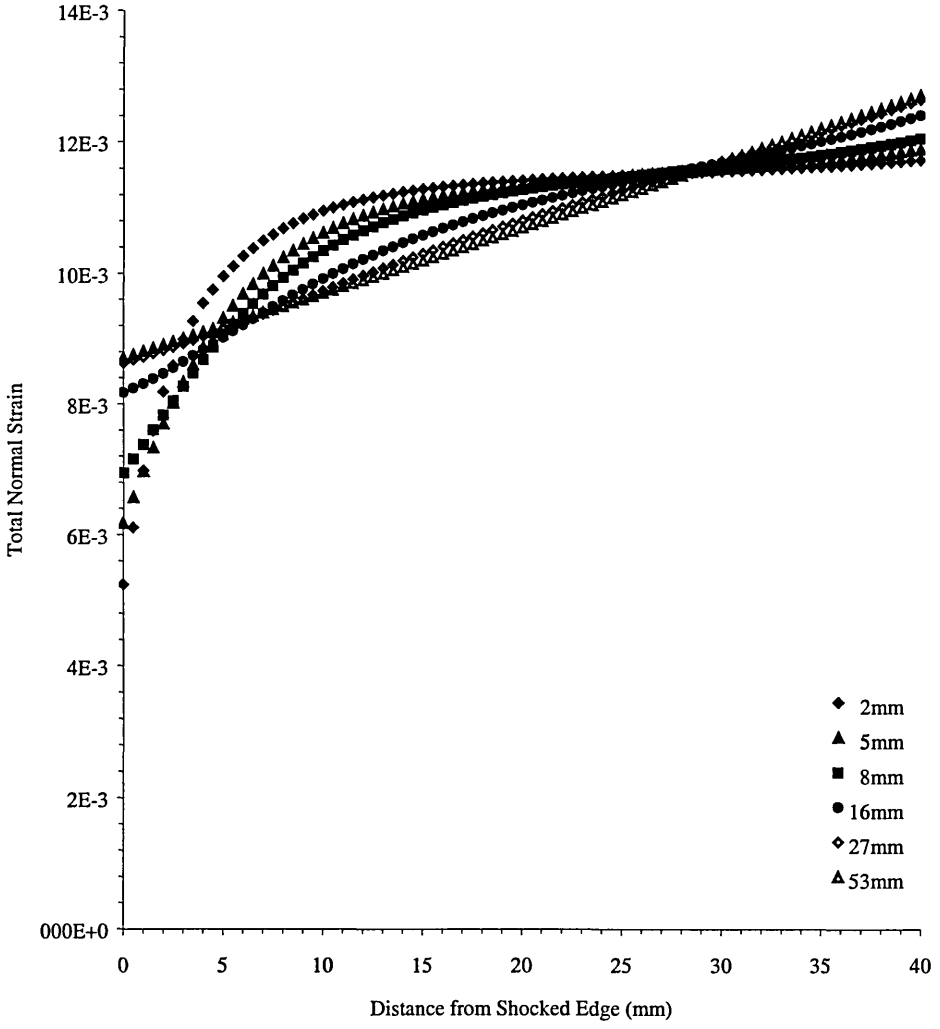


Figure 81: Effect of localisation on normal strains for 625-225°C in 3s shock by FE.

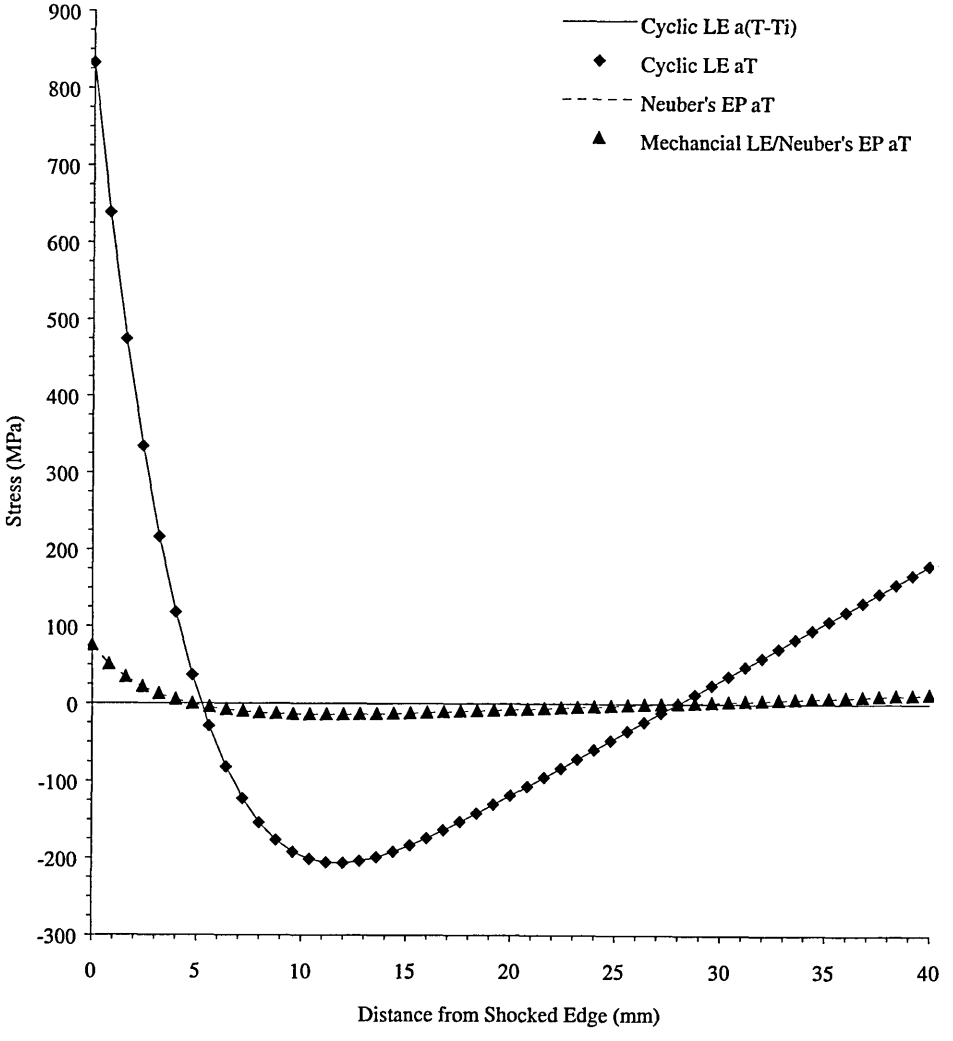


Figure 82: Elastic-plastic stresses for 625-225°C in 3s for αT strain by FD&T.

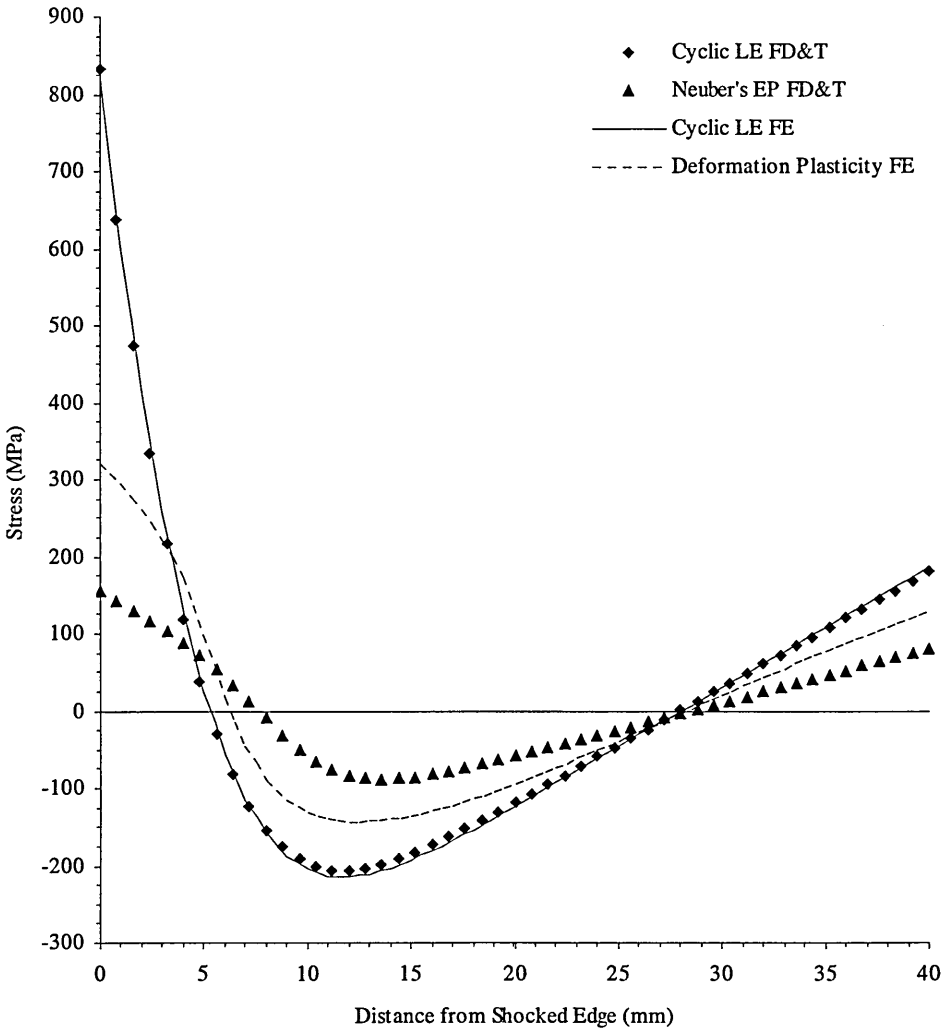


Figure 83: Elastic-plastic stresses for 625-225°C in 3s for $\alpha[T-T_i]$ strain, using Neubers full elastic-plastic material model.

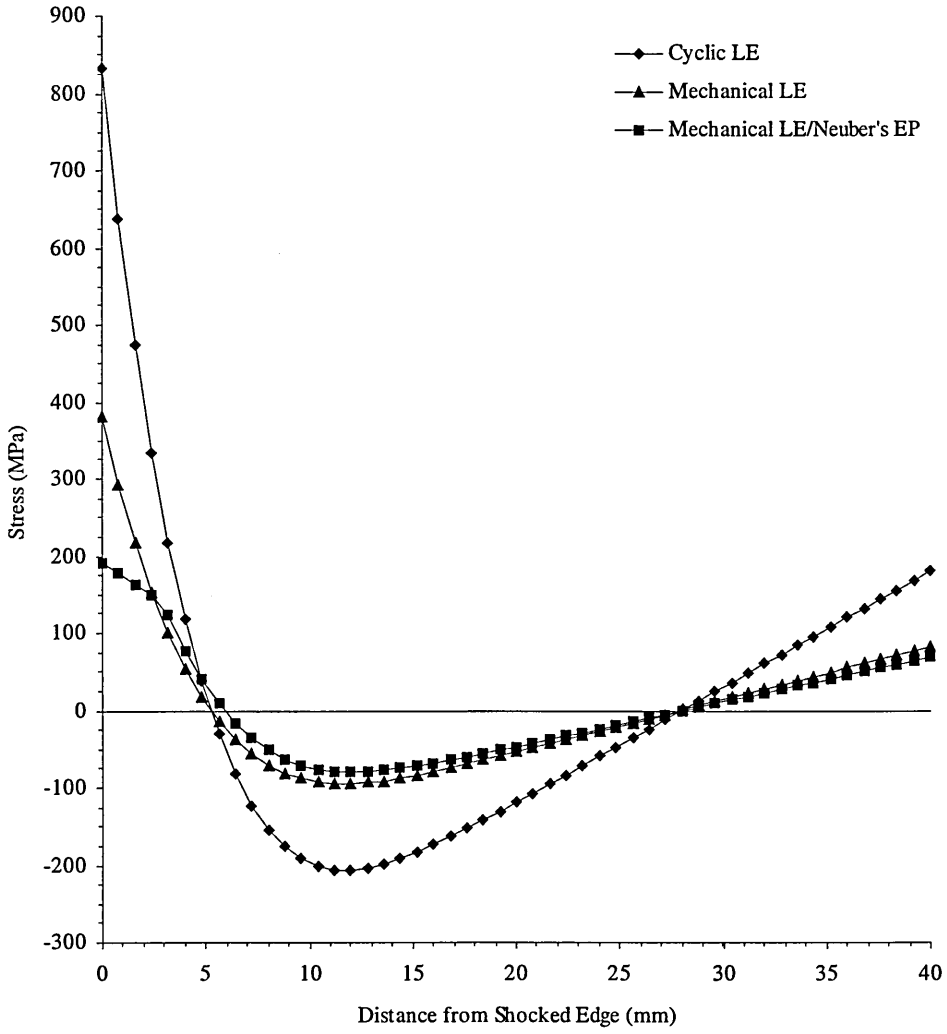


Figure 84: Elastic-plastic stresses for 625-225°C in 3s for $\alpha[T-T_i]$ strain, using partial linear-elastic elastic-plastic material model by FD&T.

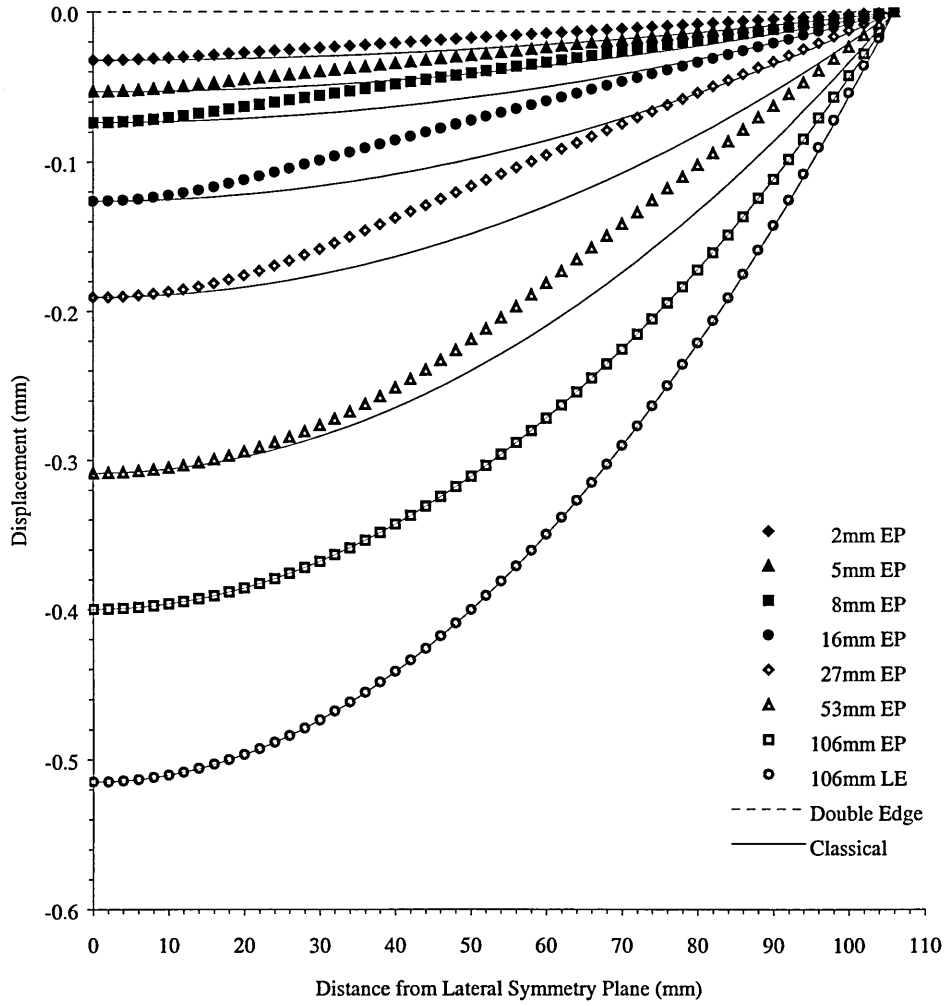


Figure 85: Single-edge displacements of centre-line for 625-225°C in 3s by FE & classical bending.

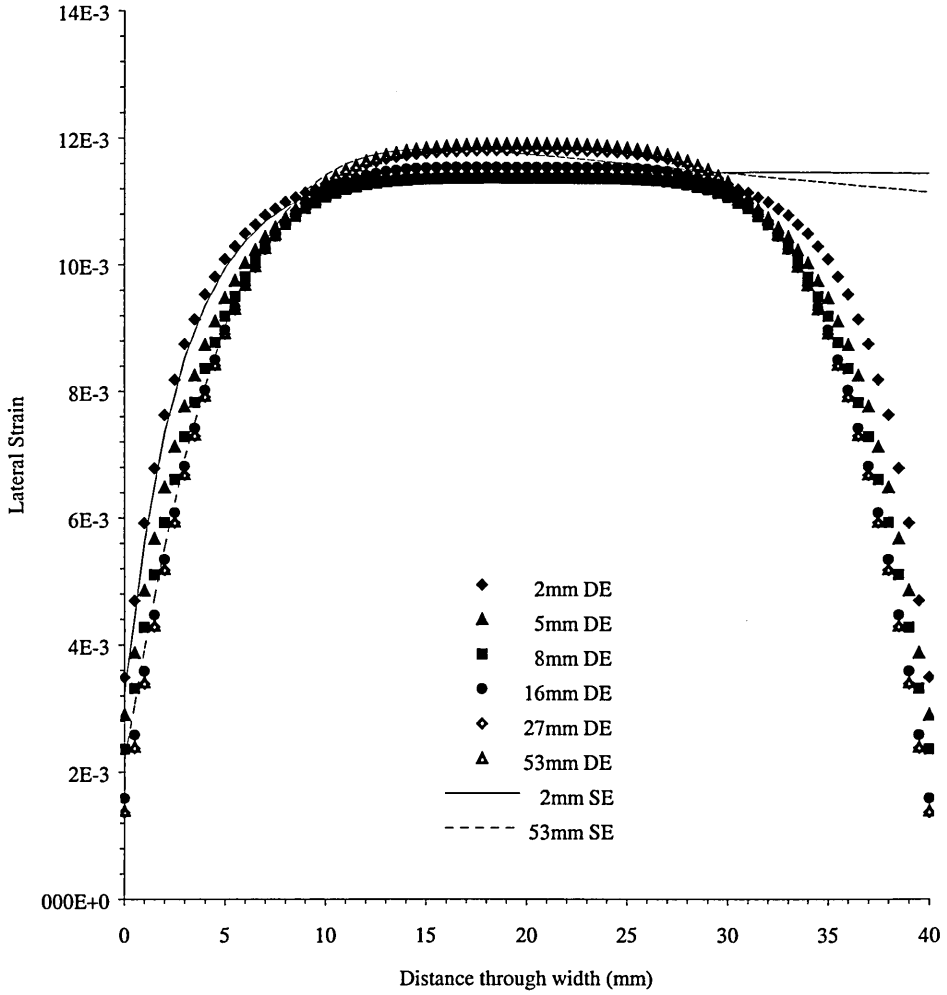


Figure 86: Double-edge lateral strain for 625-225°C in 3s by FE.

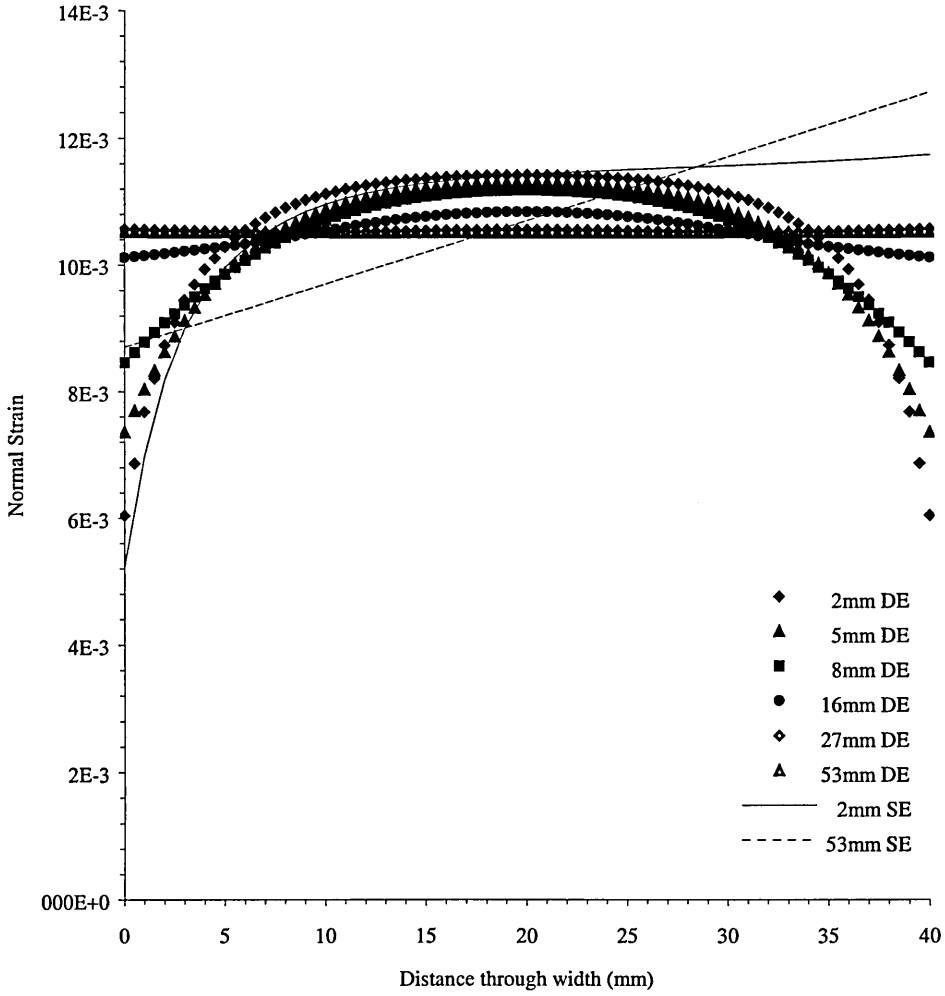


Figure 87: Double-edge normal strains for 625-225°C in 3s by FE.

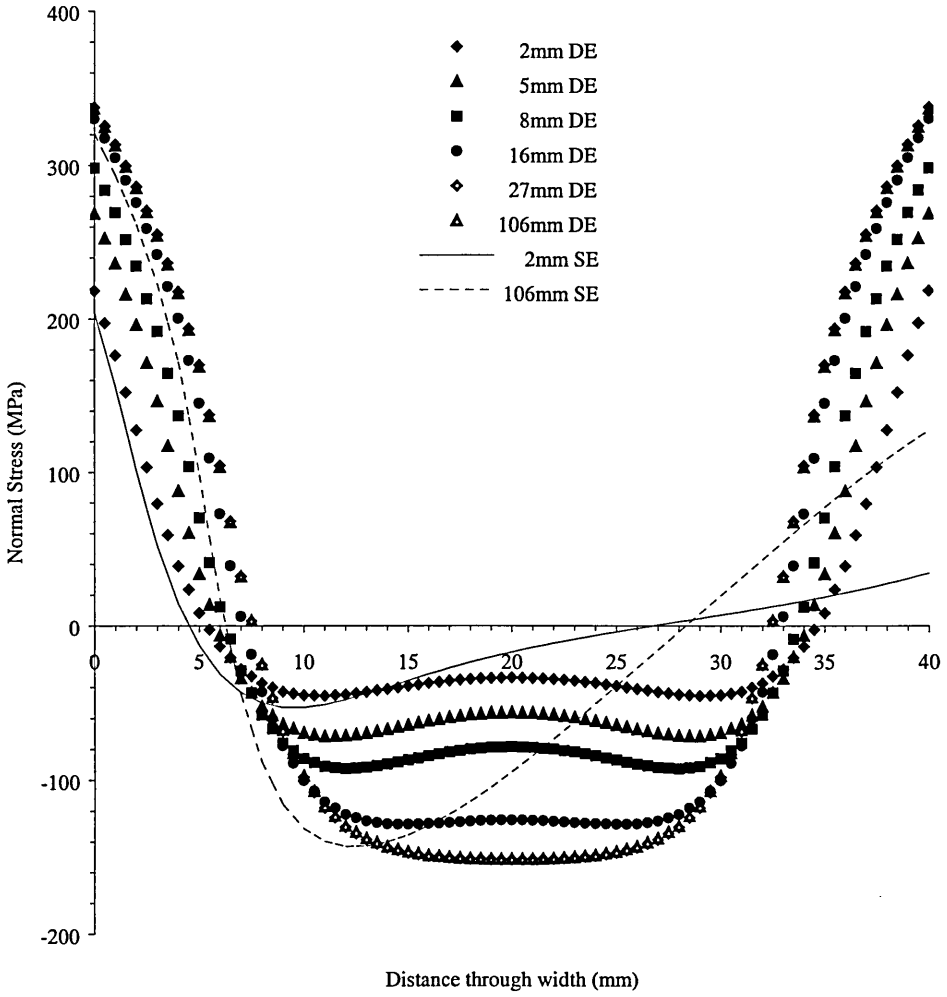


Figure 88: Double-edge normal stresses with deformation plasticity for 625-225°C in 3s by FE.

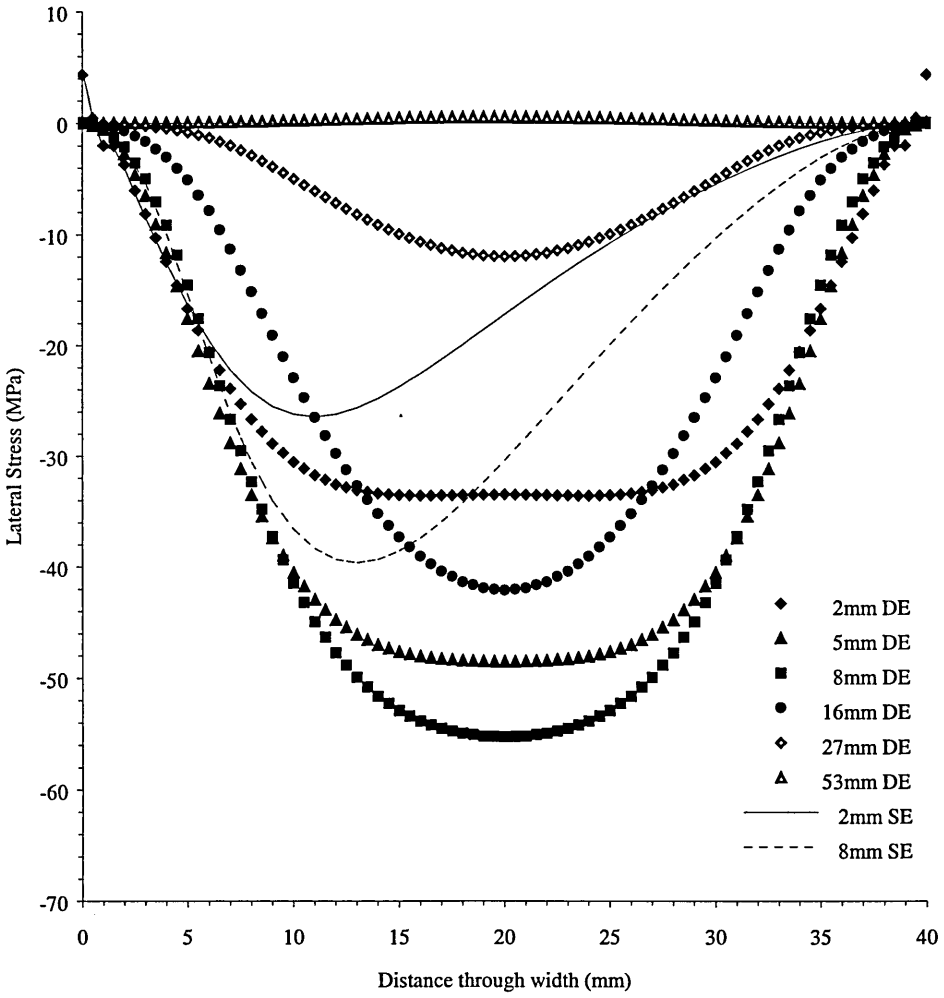


Figure 89: Double-edge lateral stresses with deformation plasticity for 625-225°C in 3s by FE.

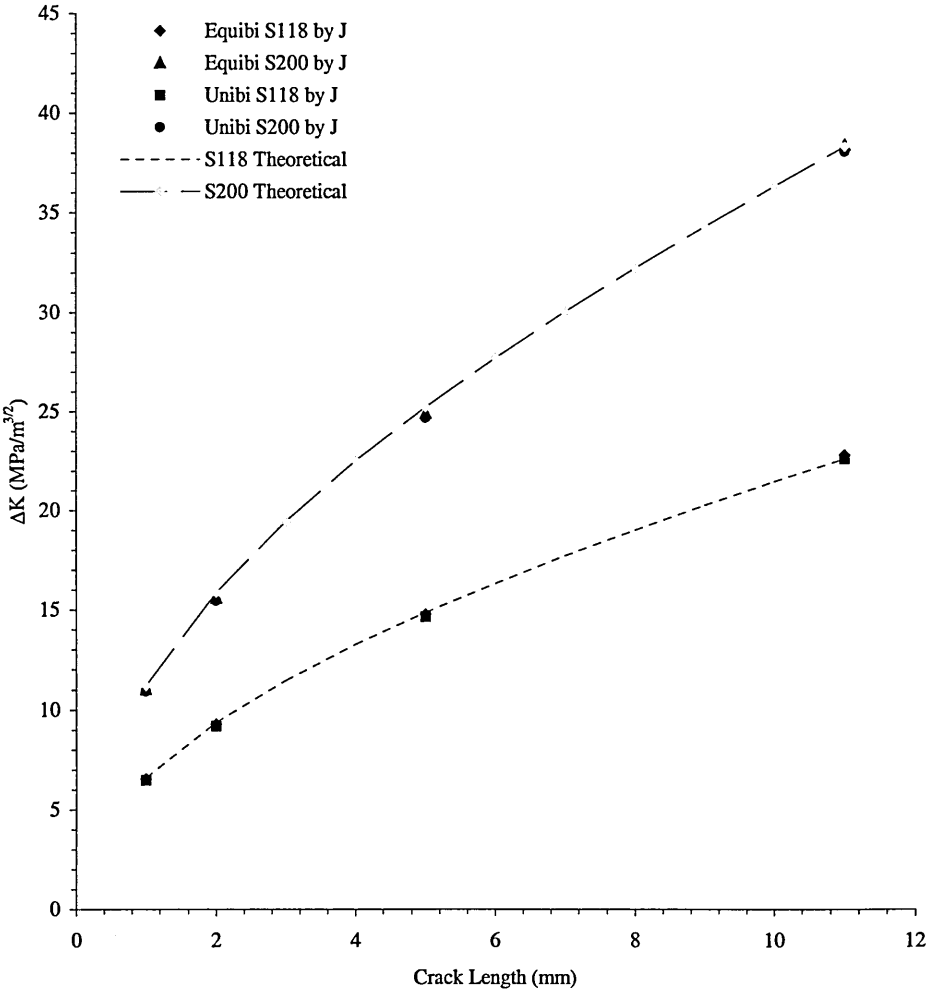


Figure 90: Linear-Elastic ΔK_I values for 4 separate loading conditions with uniform thickness of 4mm for finite element and theoretical solution.

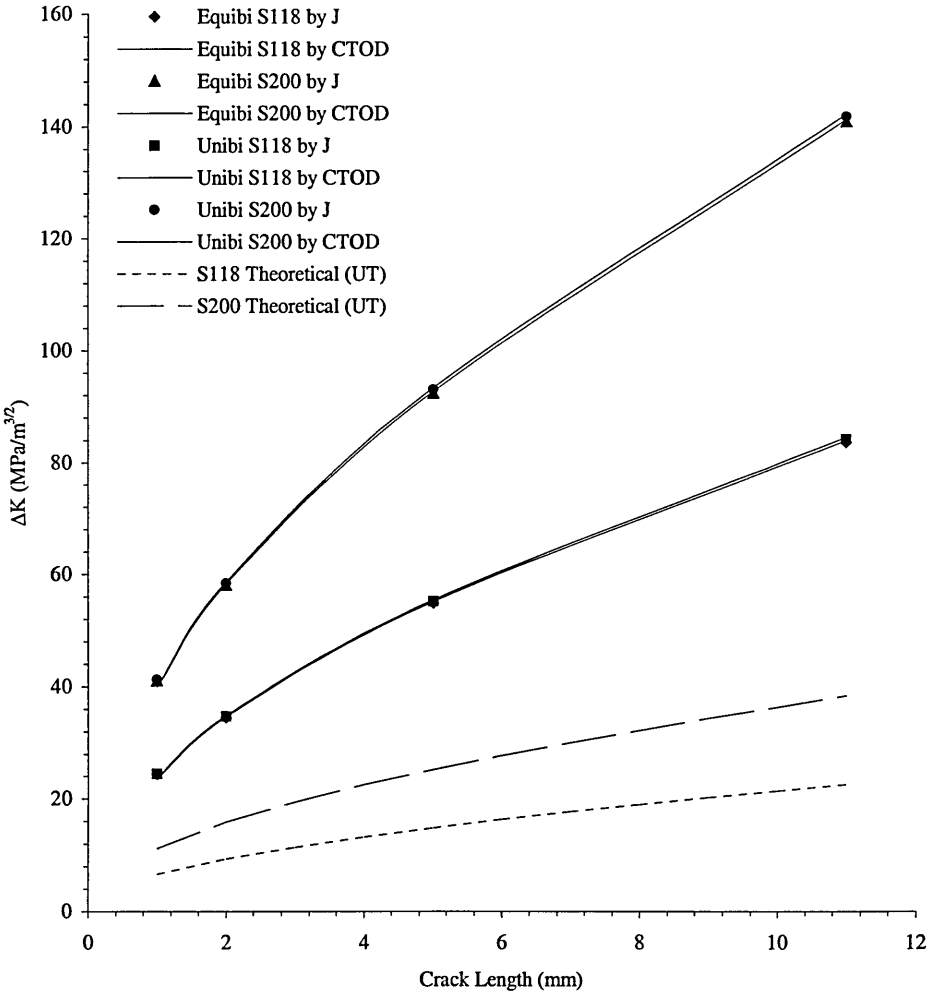


Figure 91: Linear-Elastic ΔK_I values for 4 separate loading conditions with variable thickness of 4mm at plate & 15mm at loading fins.

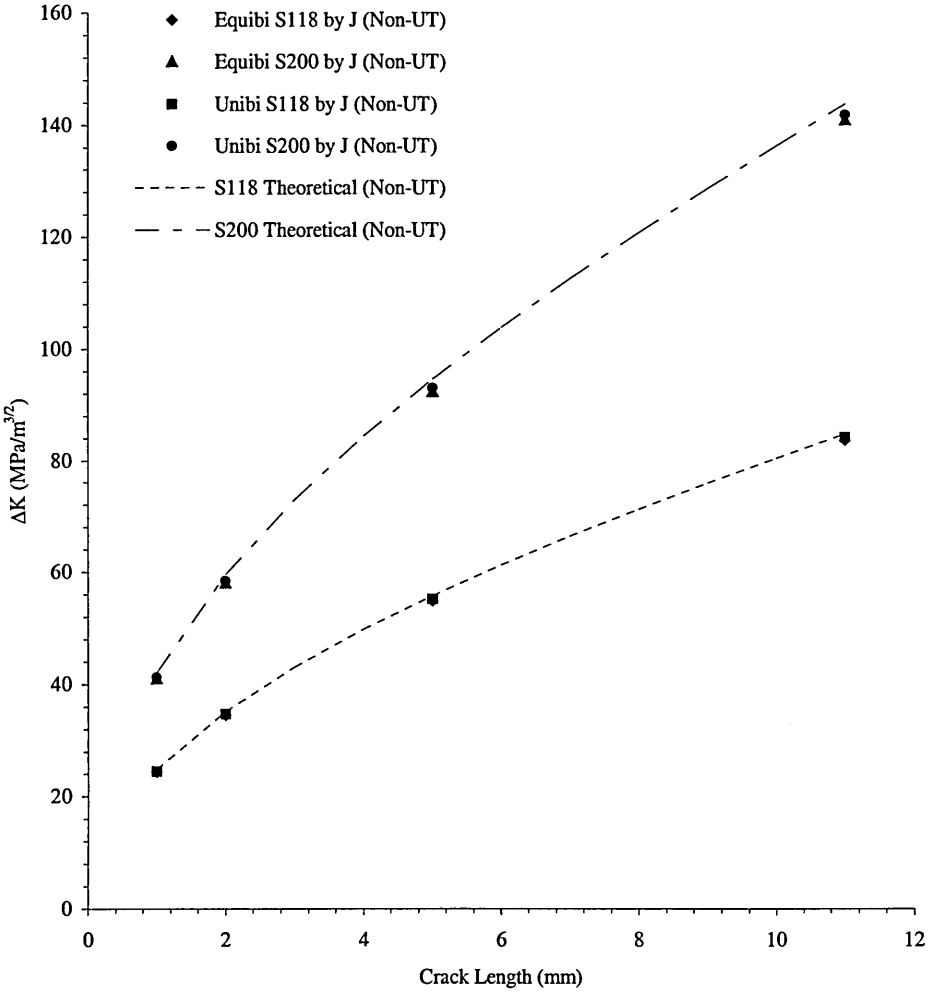


Figure 92: Modified theoretical linear-elastic ΔK_I values for 4 separate loading conditions with variable thickness of 4mm at plate & 15mm at loading fins.

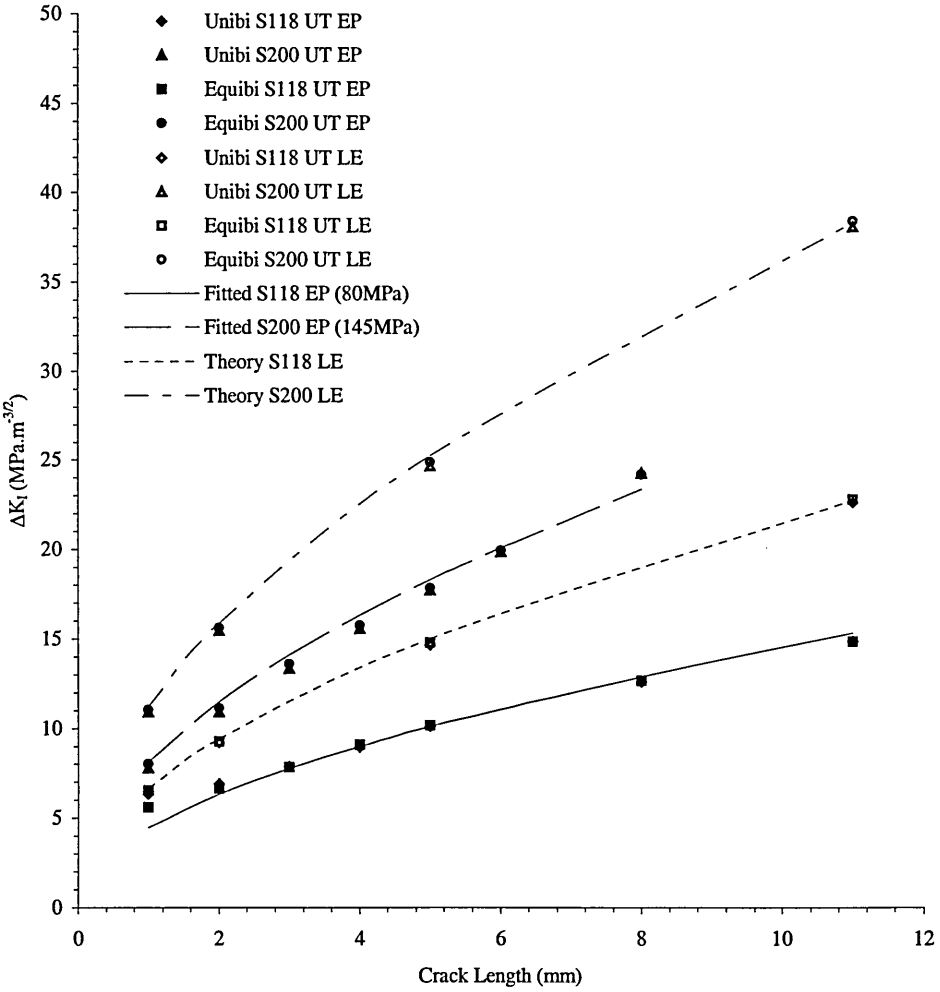


Figure 93: Elastic-Plastic & Linear-Elastic ΔK_I values for 4 separate loading conditions with uniform thickness of 4mm.

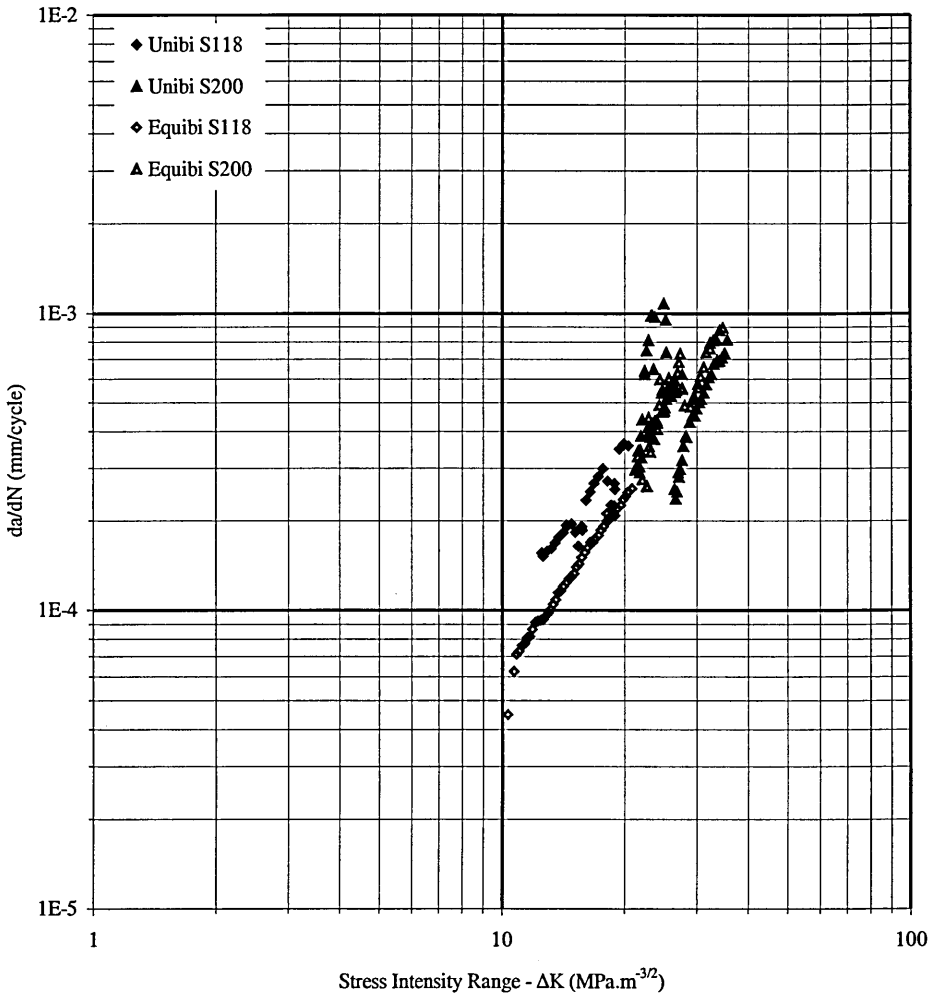


Figure 94: Isothermal cruciform crack growth rates for linear-elastic material model with stress intensity factor for uniform thickness (experimental [124]/FE).

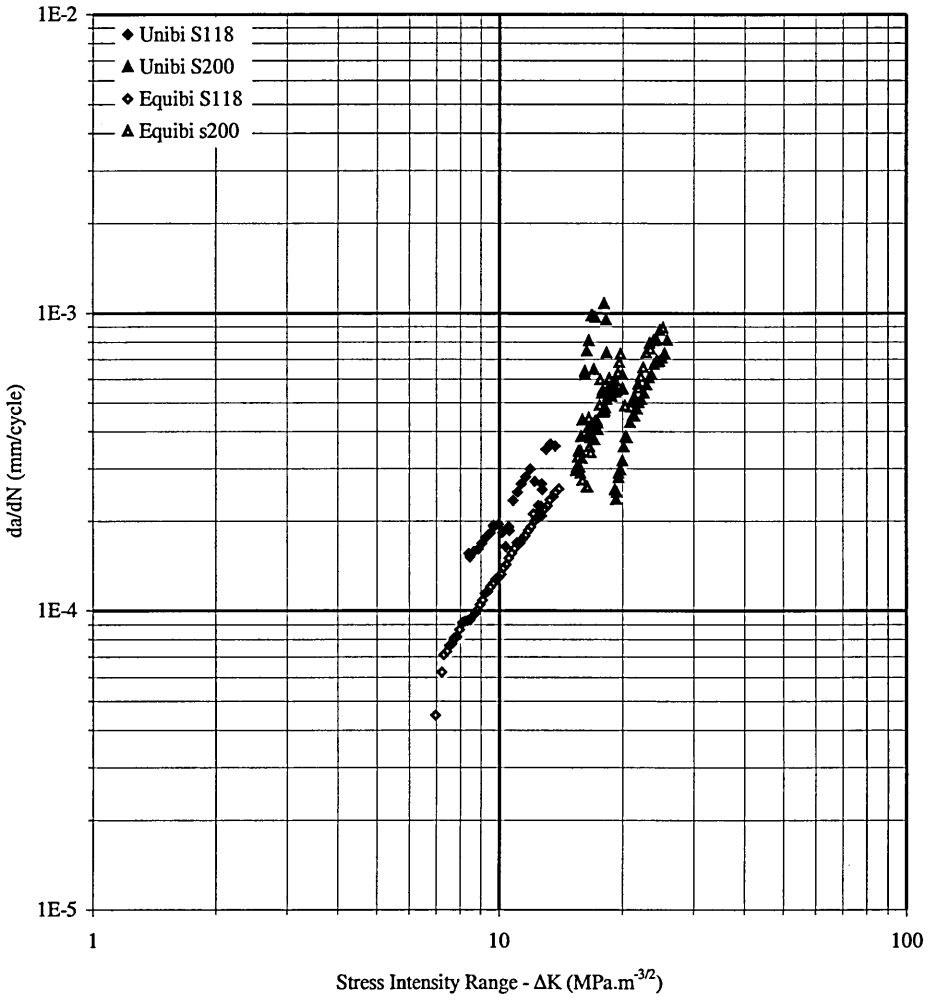


Figure 95: Isothermal cruciform crack growth rates for elastic-plastic material model with stress intensity factor for uniform thickness (experimental [124]/FE).

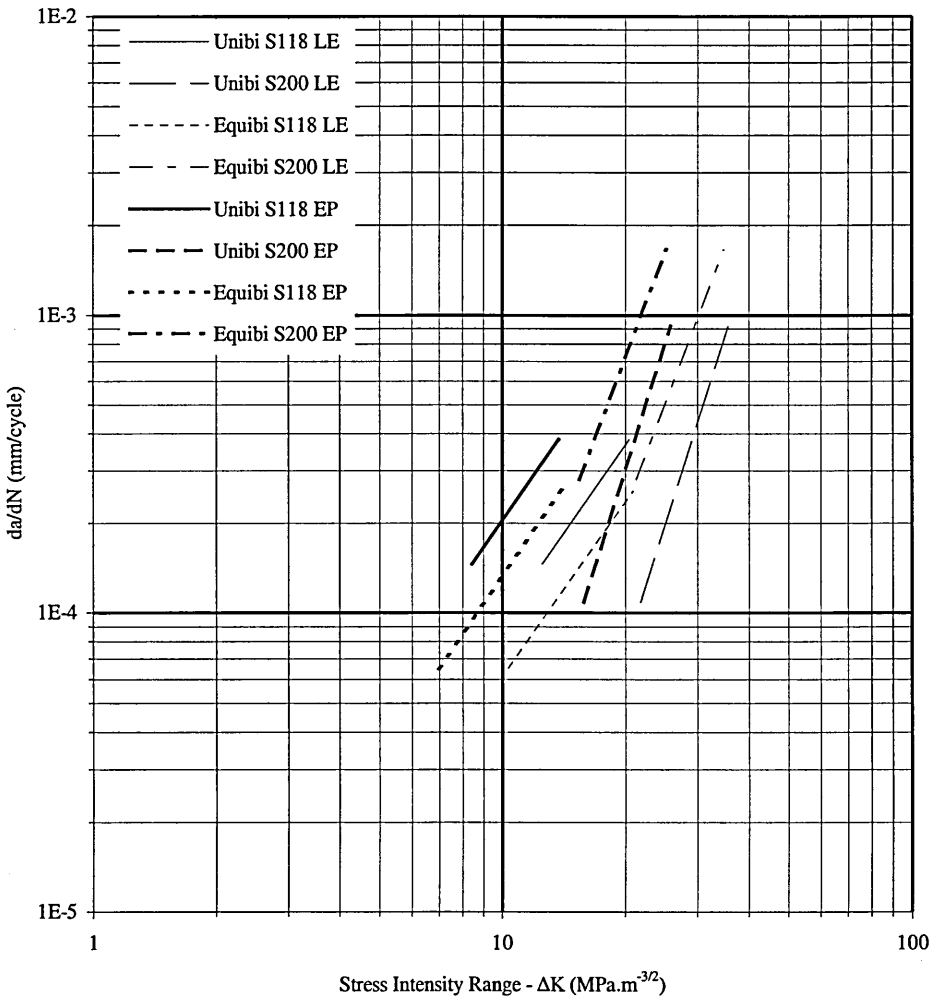


Figure 96: Comparison of isothermal cruciform crack growth rates for linear-elastic & elastic-plastic material model with stress intensity factor for uniform thickness (experimental [124]/FE).

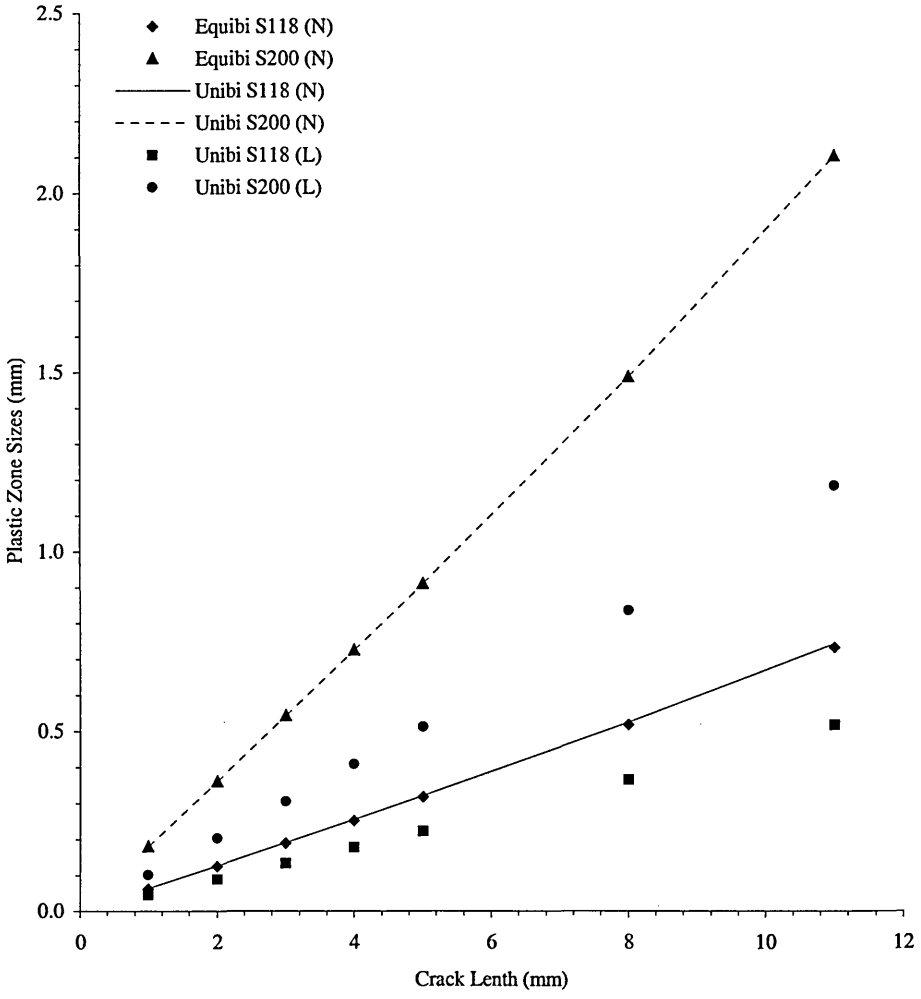


Figure 97: Theoretical linear-elastic normal (N) & lateral (L) component plastic zone sizes.

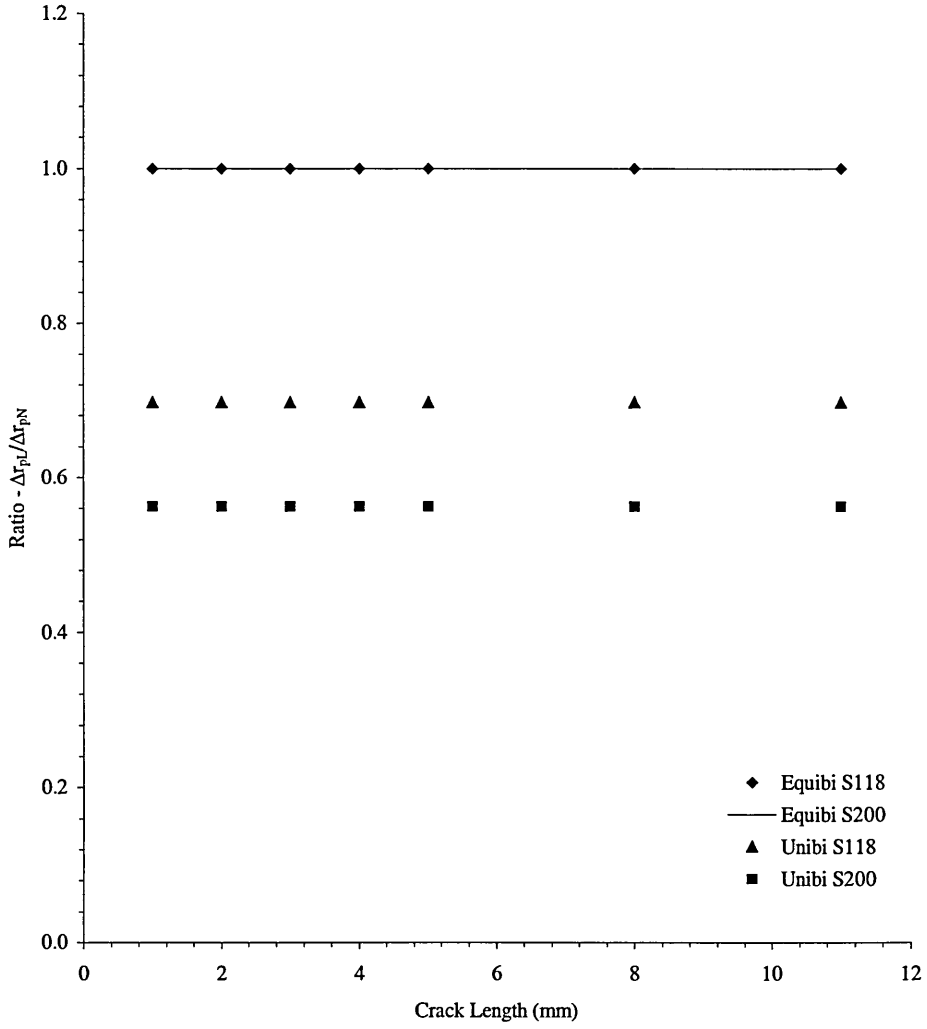


Figure 98: Ratio of theoretical linear-elastic change in lateral to normal component plastic zones.

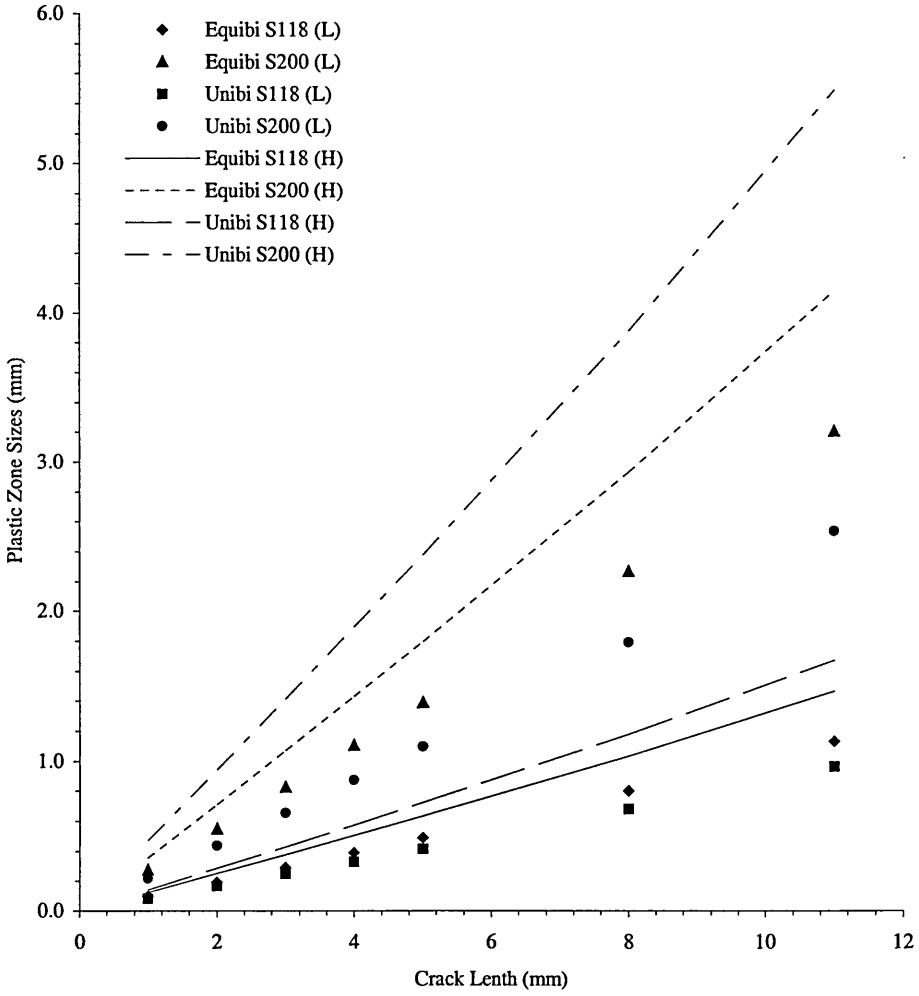


Figure 99: Length (L) & height (H) of theoretical linear-elastic von Mises plastic zone accounting for T-Stress, σ_T.

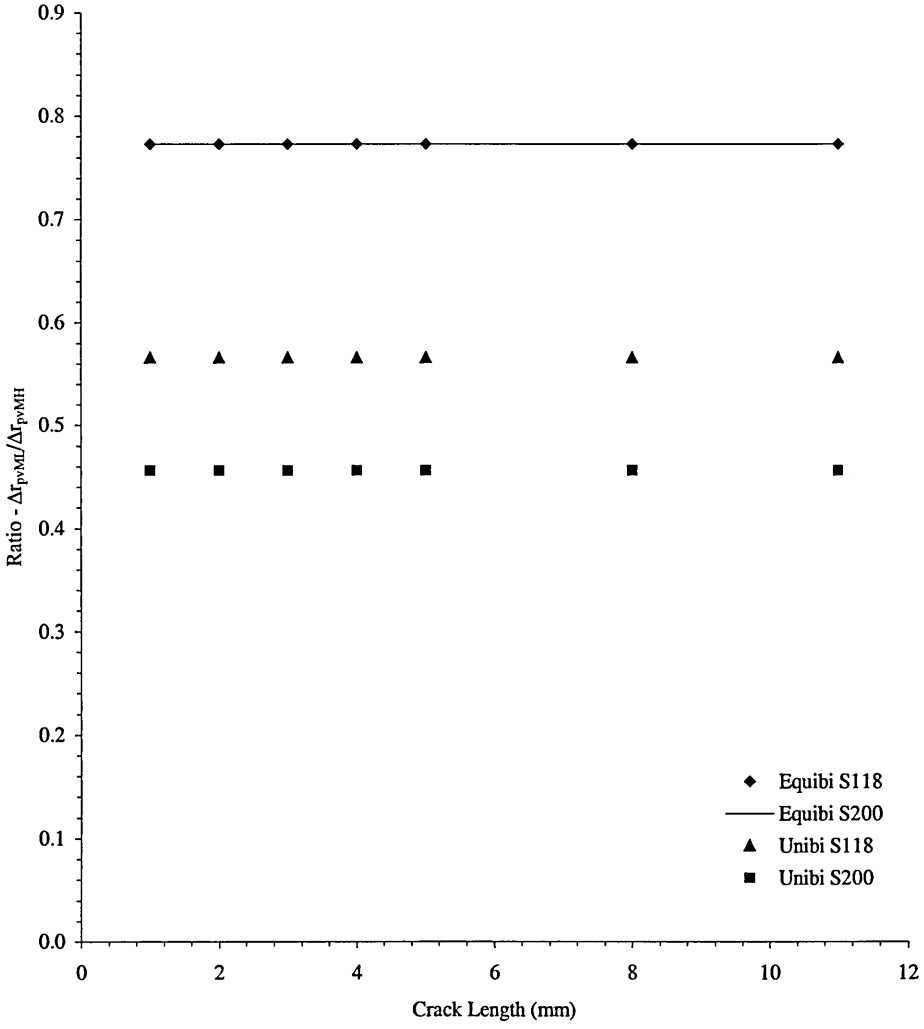


Figure 100: Ratio of change in length (L) & height (H) of theoretical linear-elastic von Mises plastic zones accounting for T-Stress, σ_T.

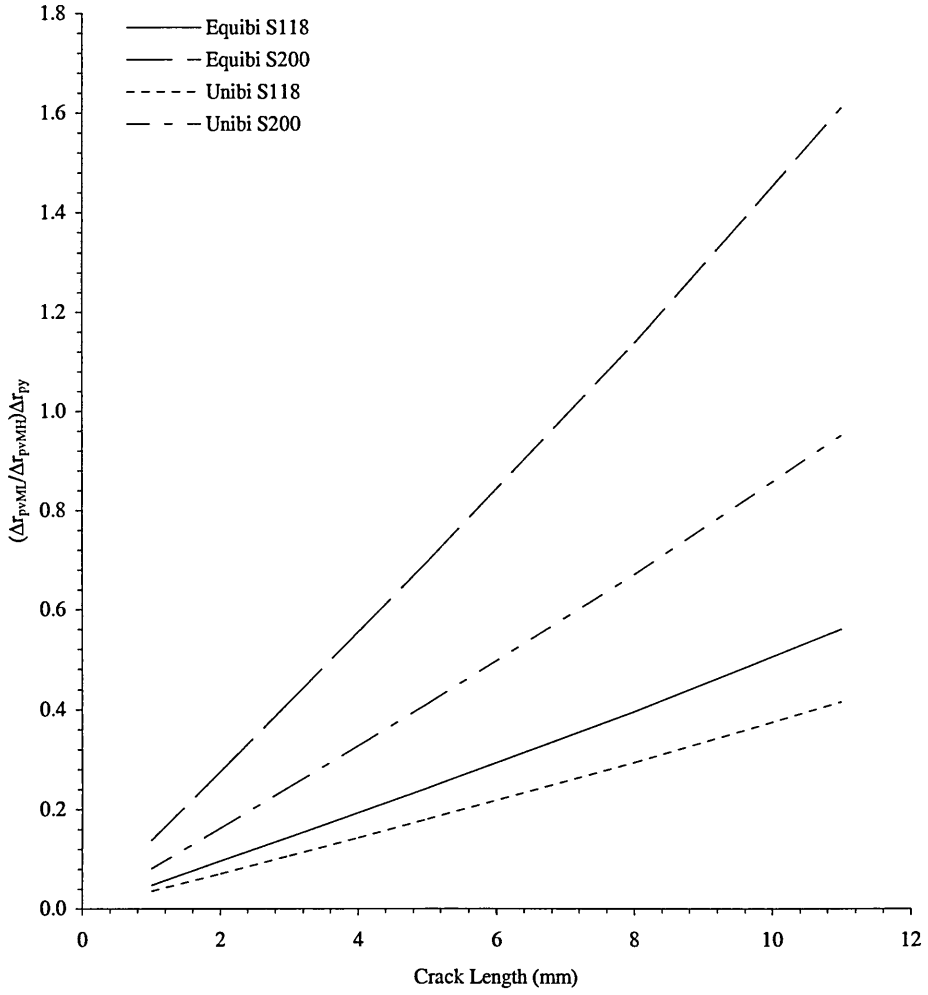


Figure 101: Biaxial plastic-zone parameter for theoretical linear-elastic material accounting for T-Stress, σ_T.

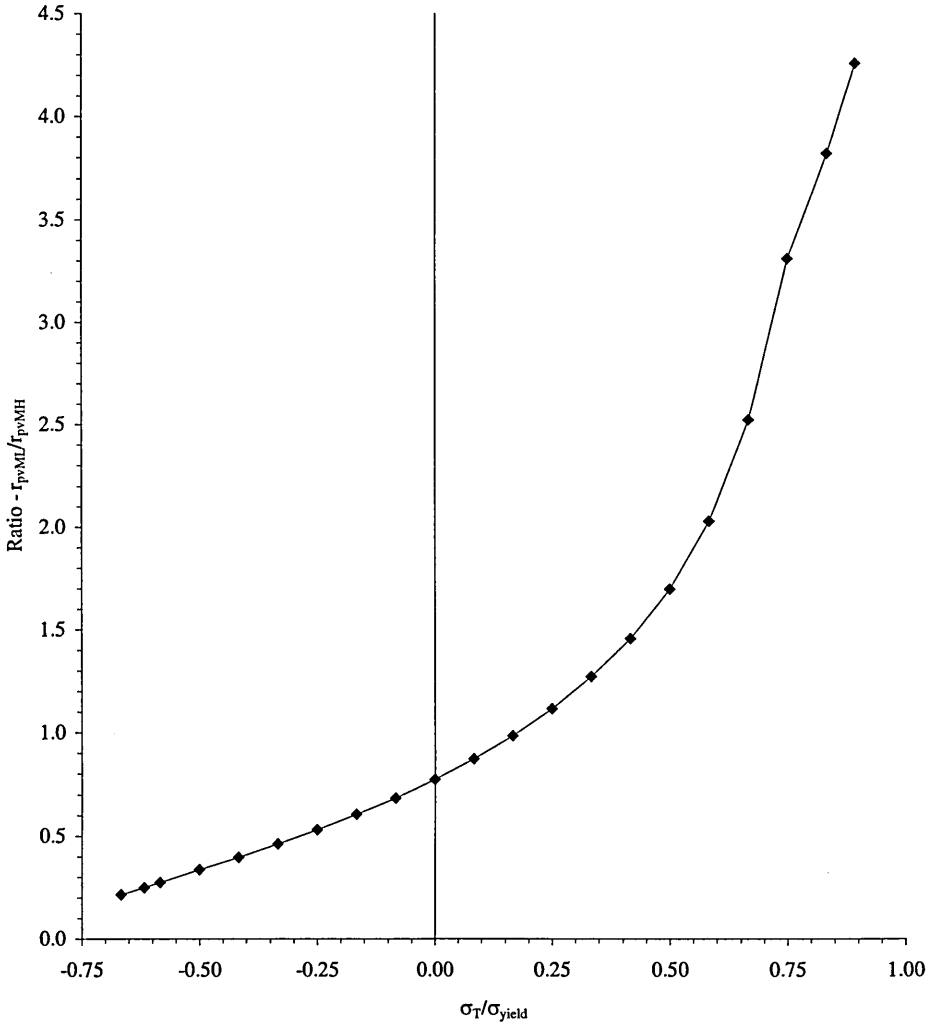


Figure 102: Effect of T-Stress on von Mises Shape Ratio.

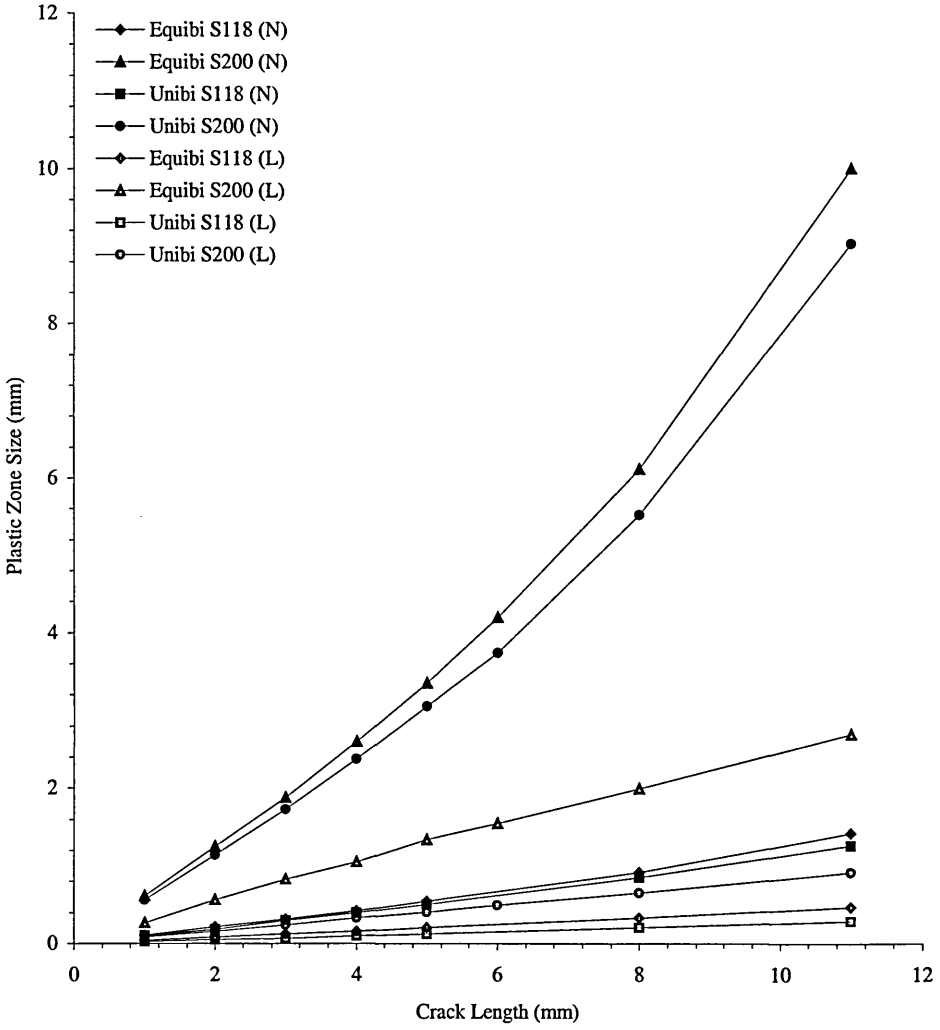


Figure 103: Normal (N) & lateral (L) plastic zone sizes at 4 isothermal biaxial loading conditions for elastic-plastic model at maximum load.

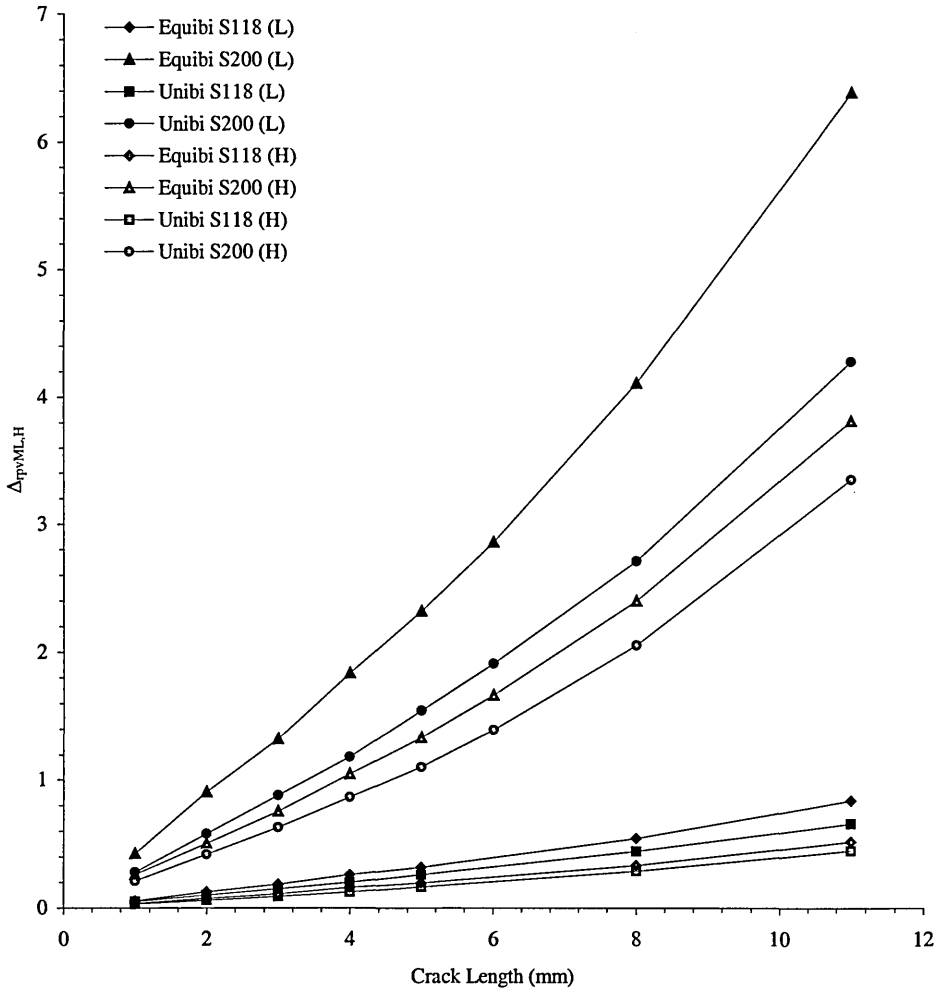


Figure 104: Change in length (L) & height (H) of von Mises plastic zones at 4 loading conditions for elastic-plastic model.

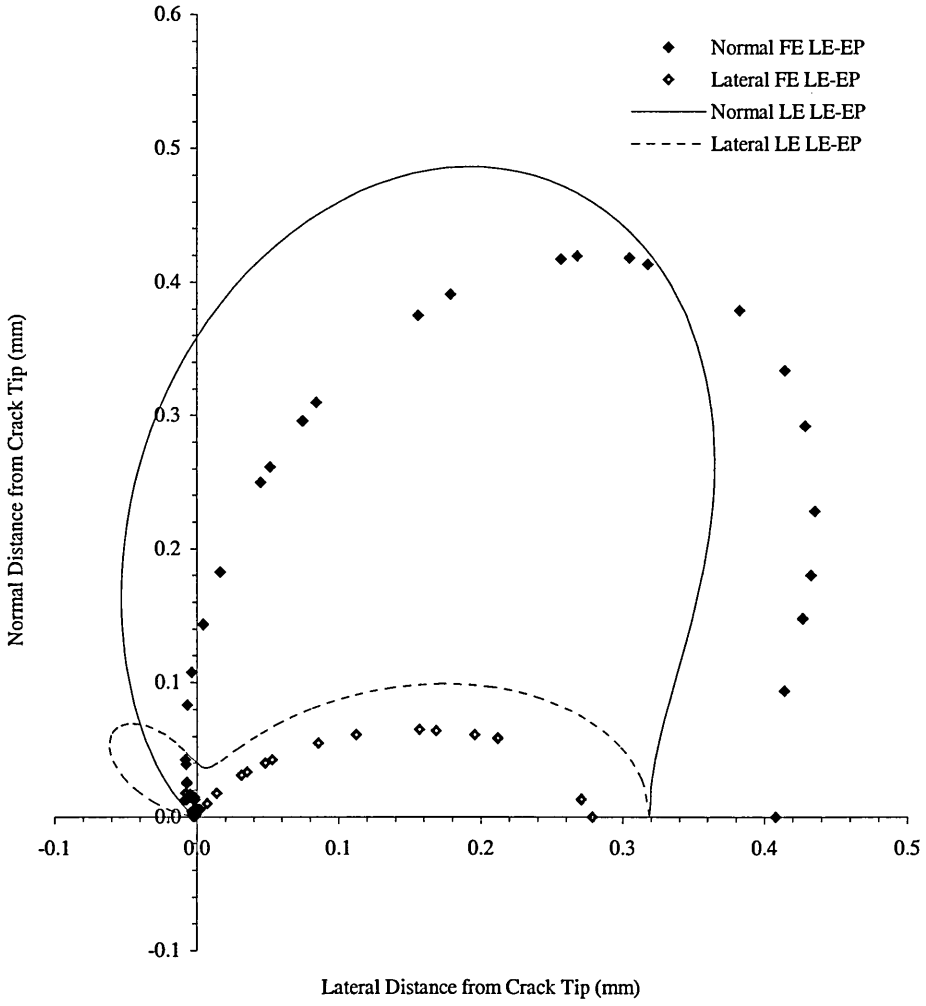


Figure 105: Effect of plasticity on normal & lateral plastic zones for equibiaxial loading at a 5mm crack length.

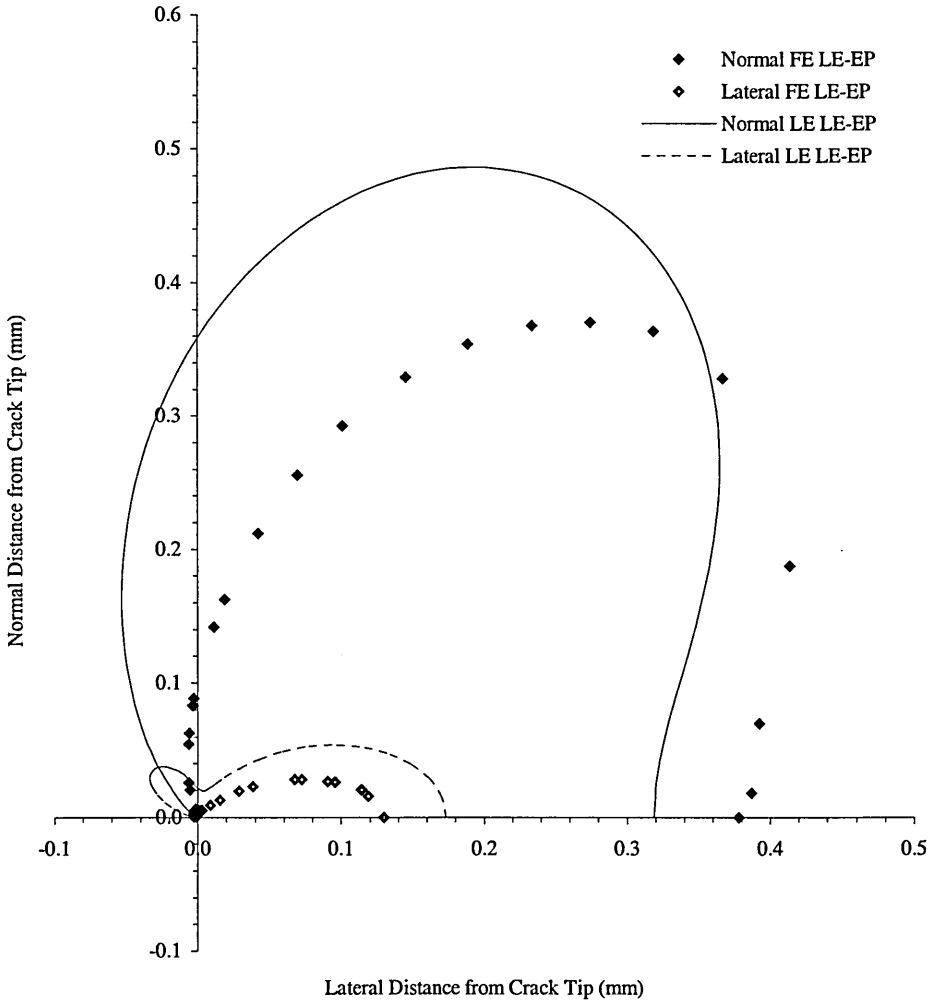


Figure 106: Effect of plasticity on normal & lateral plastic zones for unibiaxial loading at a 5mm crack length.

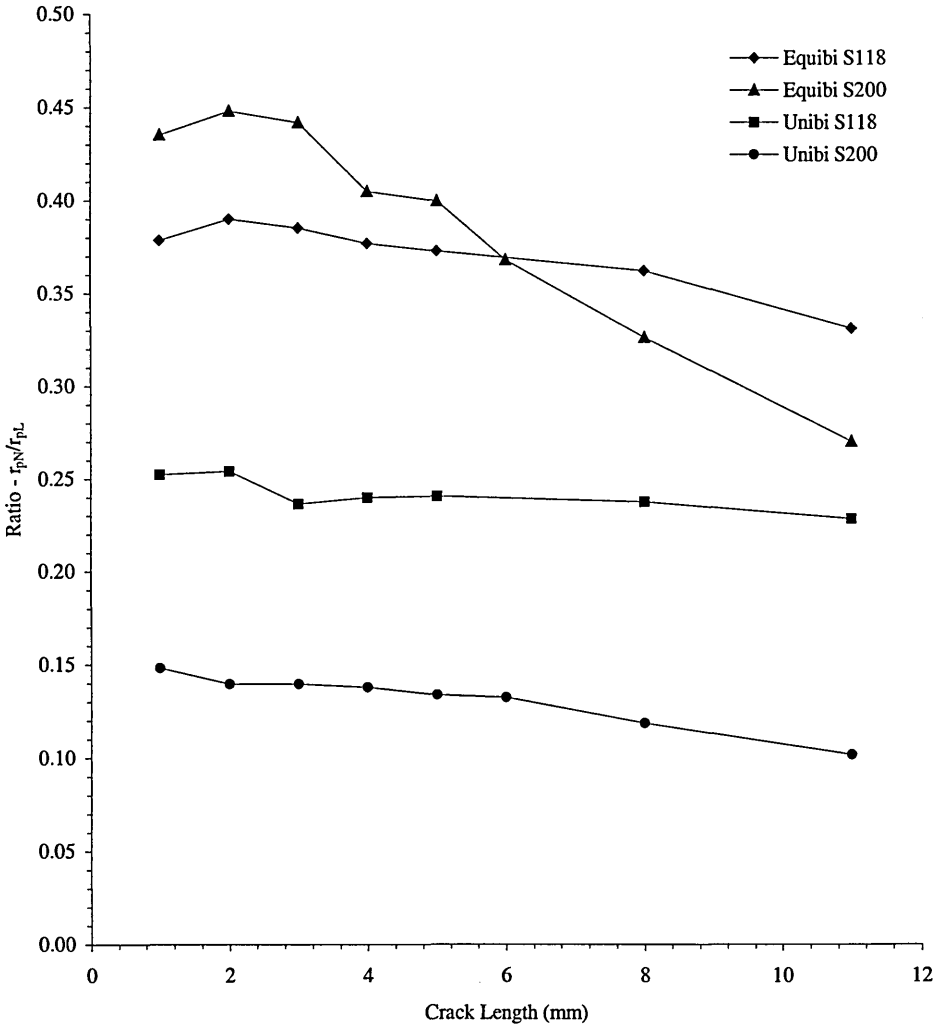


Figure 107: Ratio of lateral to normal plastic zone sizes for elastic-plastic model at maximum load.

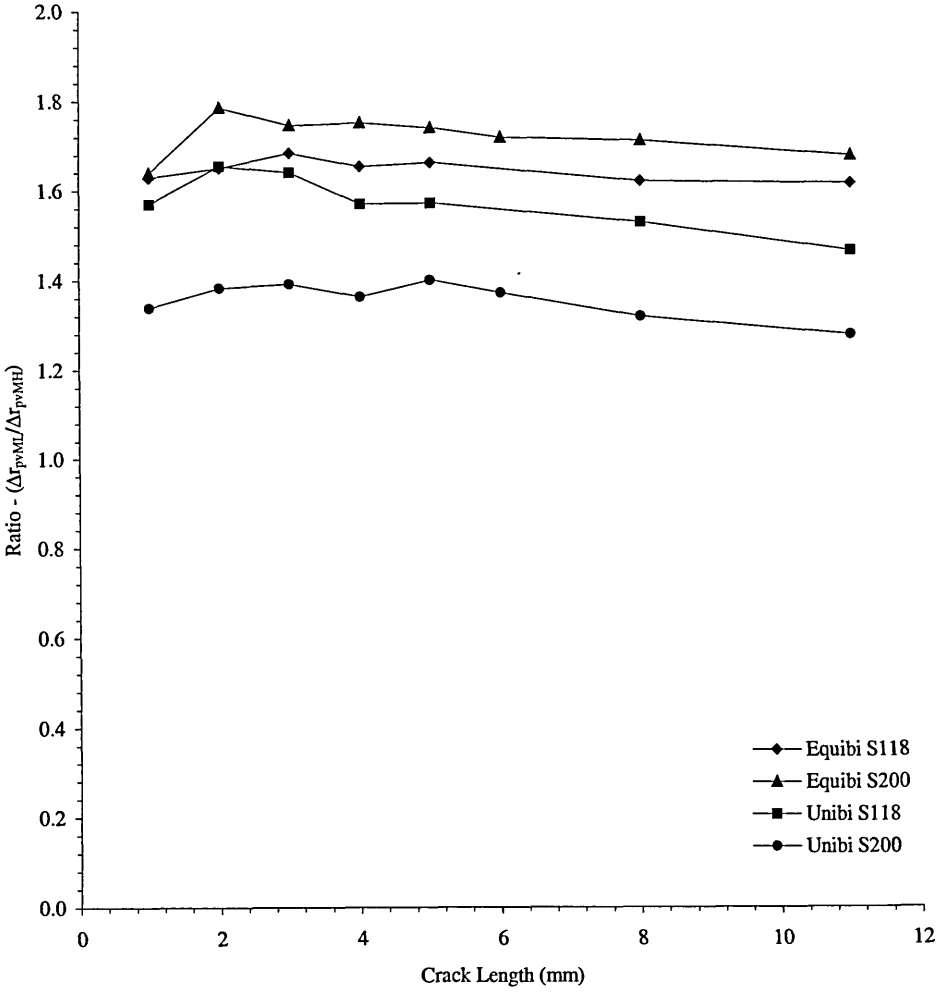


Figure 108: Ratio of change in length to change in height of von Mises plastic zone.

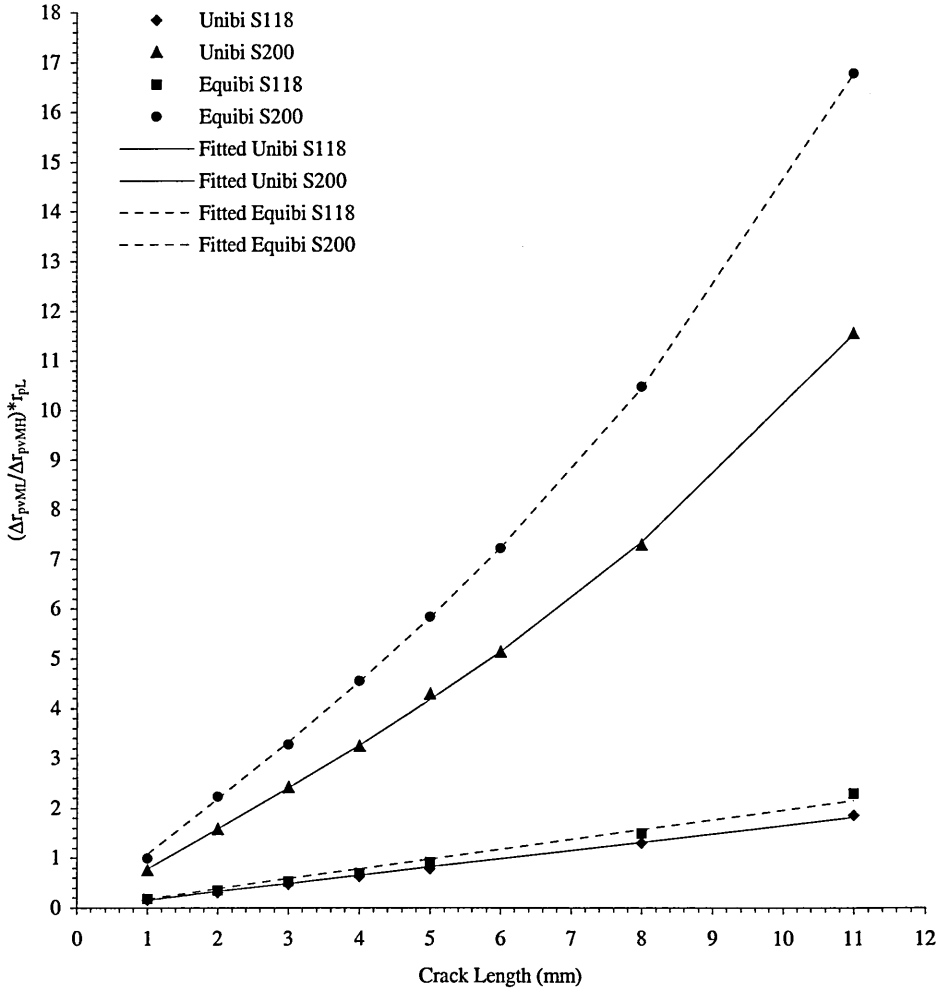


Figure 109: Biaxial von Mises plastic zone function with maximum load normal zone for LE-EP material.

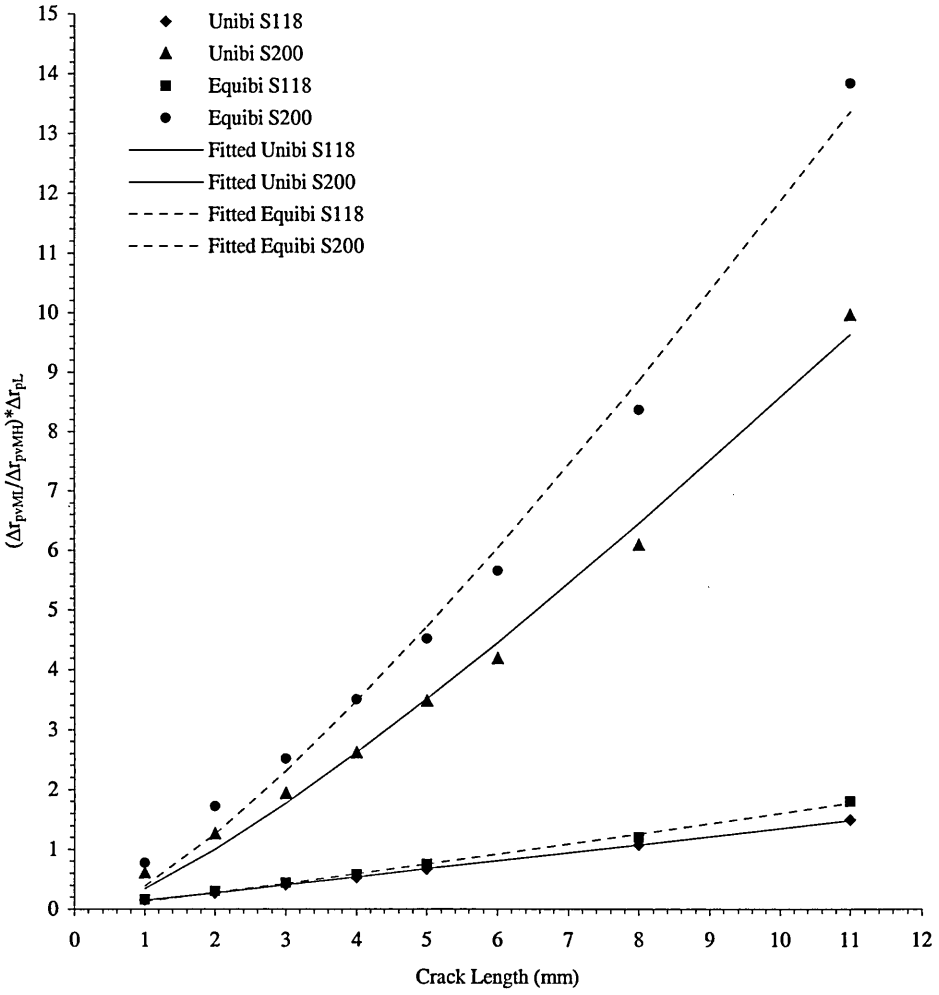


Figure 110: Biaxial von Mises plastic zone function with change in normal zone using LE minimum plastic zone from minimum LE-EP stress intensity factor.

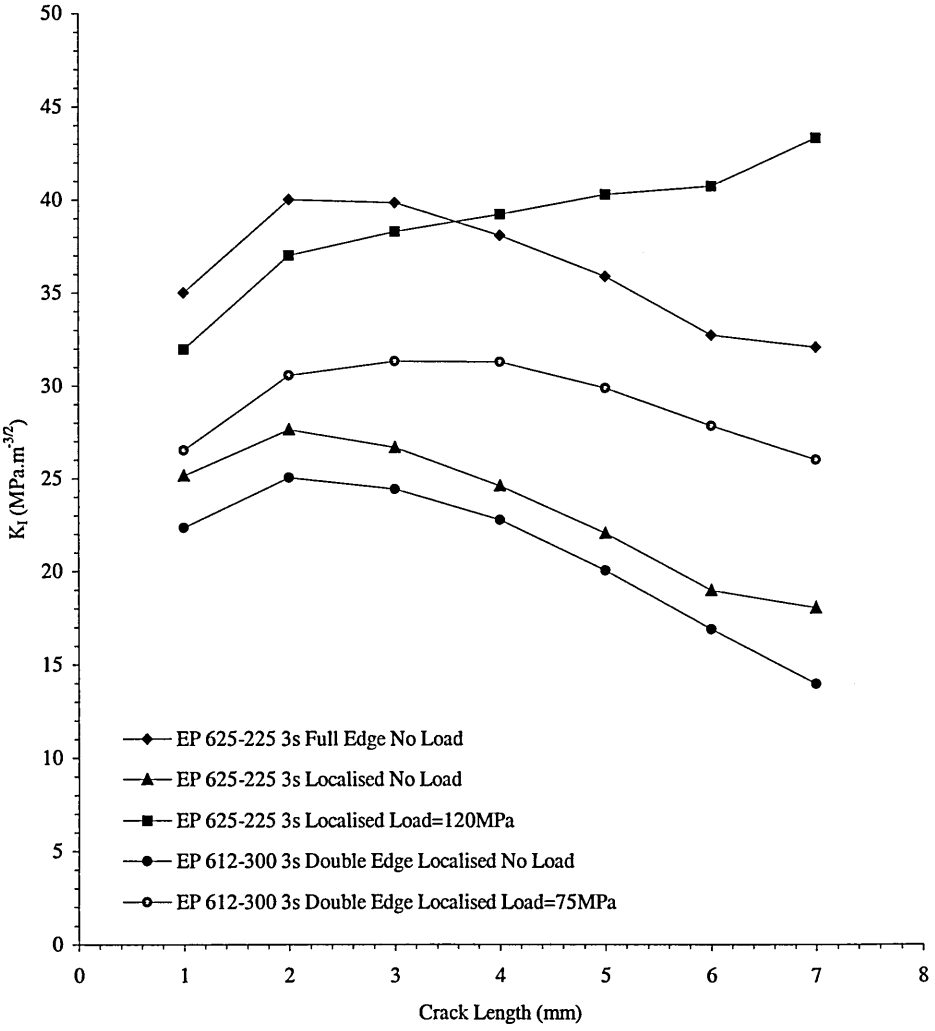


Figure 111: Thermal shock effective stress intensity factors taken from the end of the applied downshock.

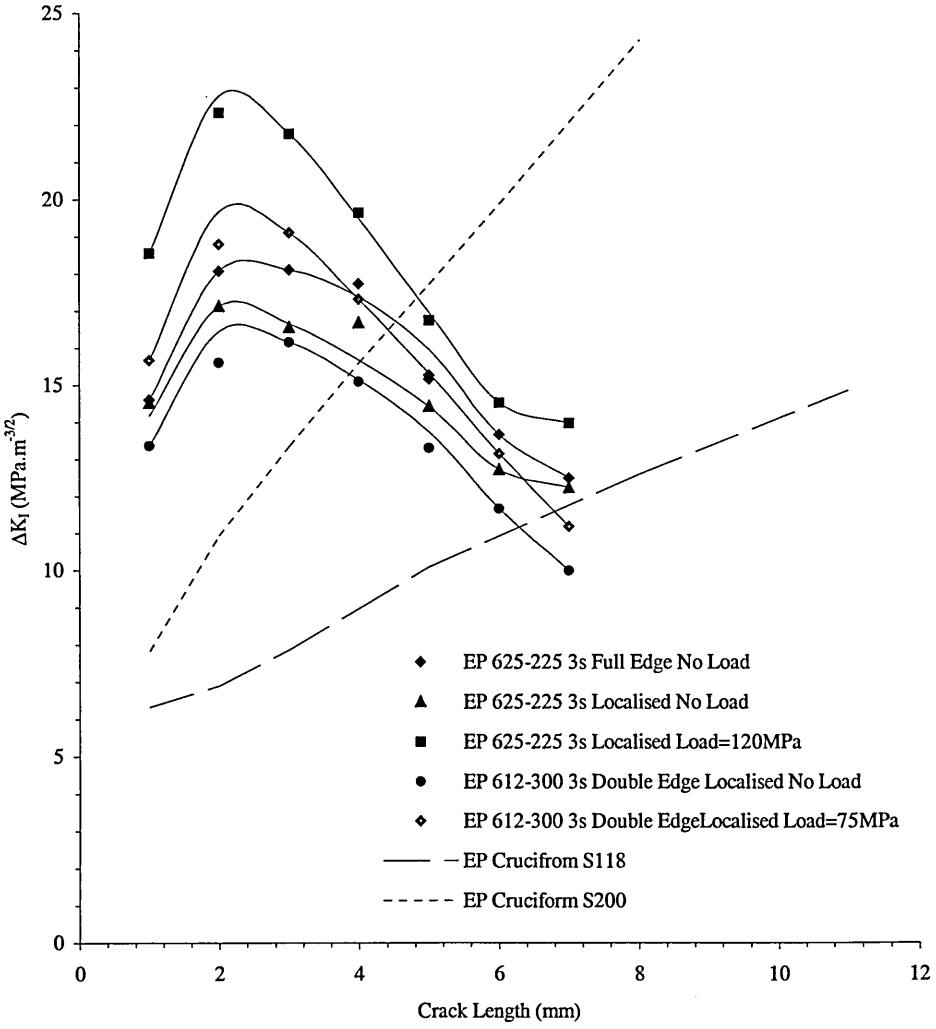


Figure 112: Thermal shock stress intensity ranges with isothermal ranges cruciform for comparison.

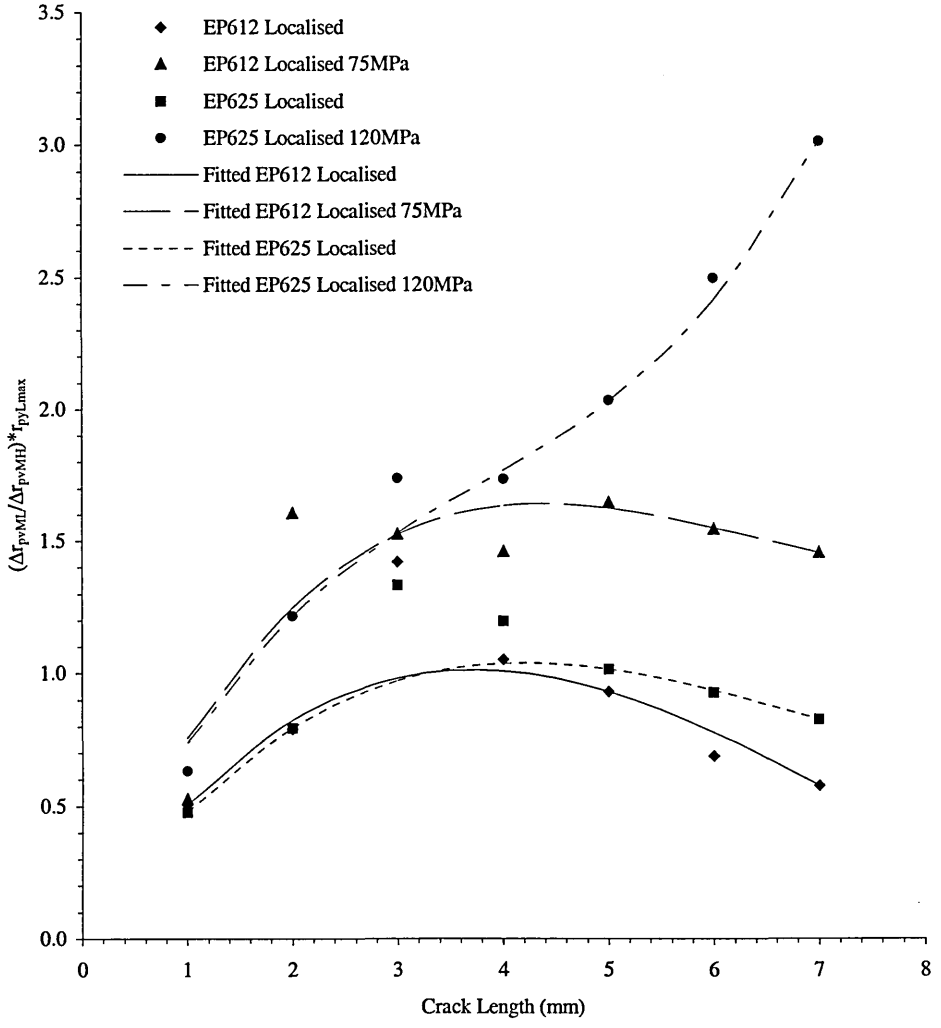


Figure 113: Plastic zone parameter for thermal shock using maximum load normal zone.

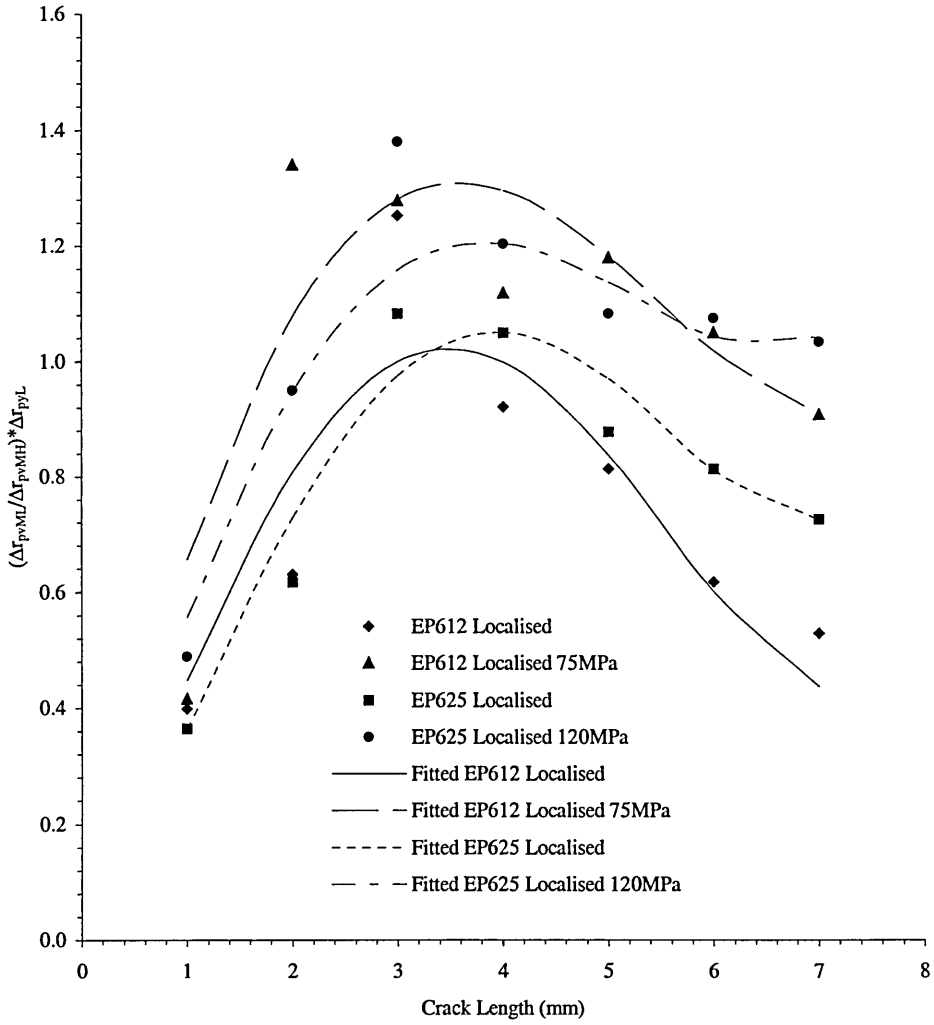


Figure 114: Plastic zone parameter for thermal shock with change in normal zone using LE minimum plastic zone from minimum LE-EP stress intensity factor.

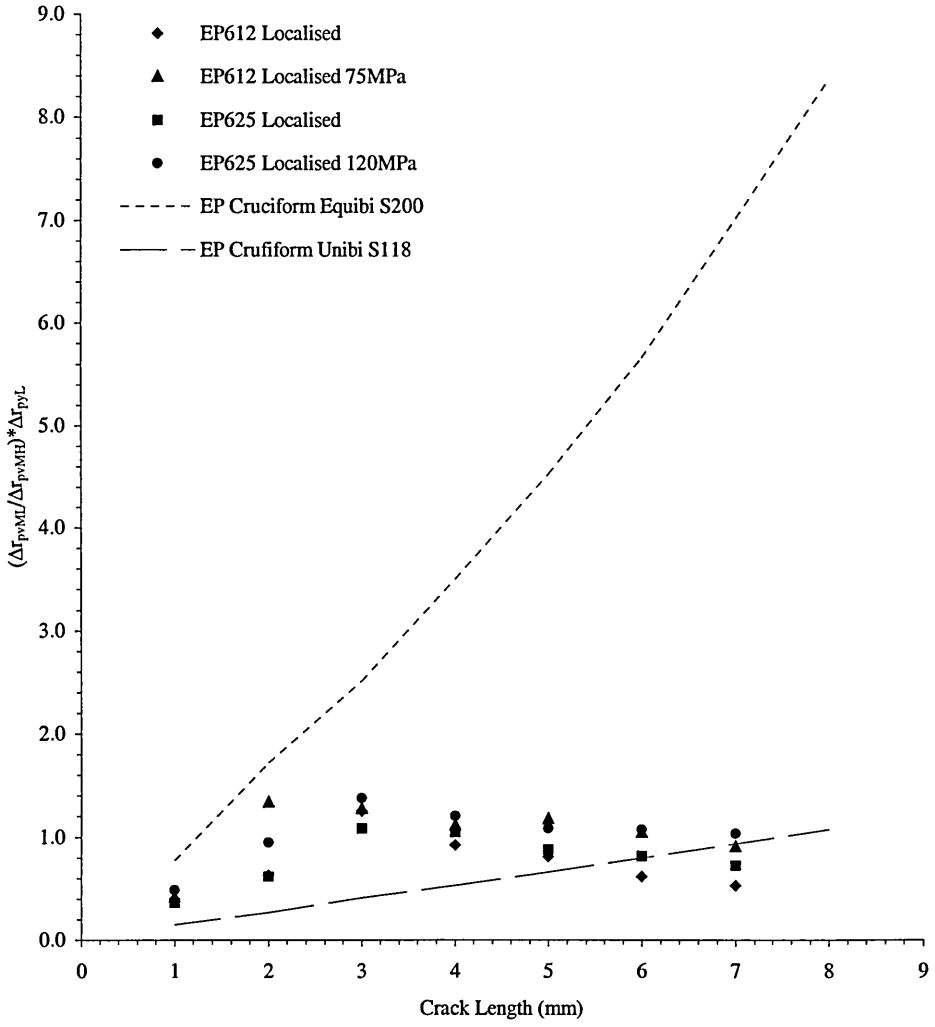


Figure 115: Comparison of plastic zone parameter for thermal shock and cruciform models by FE.

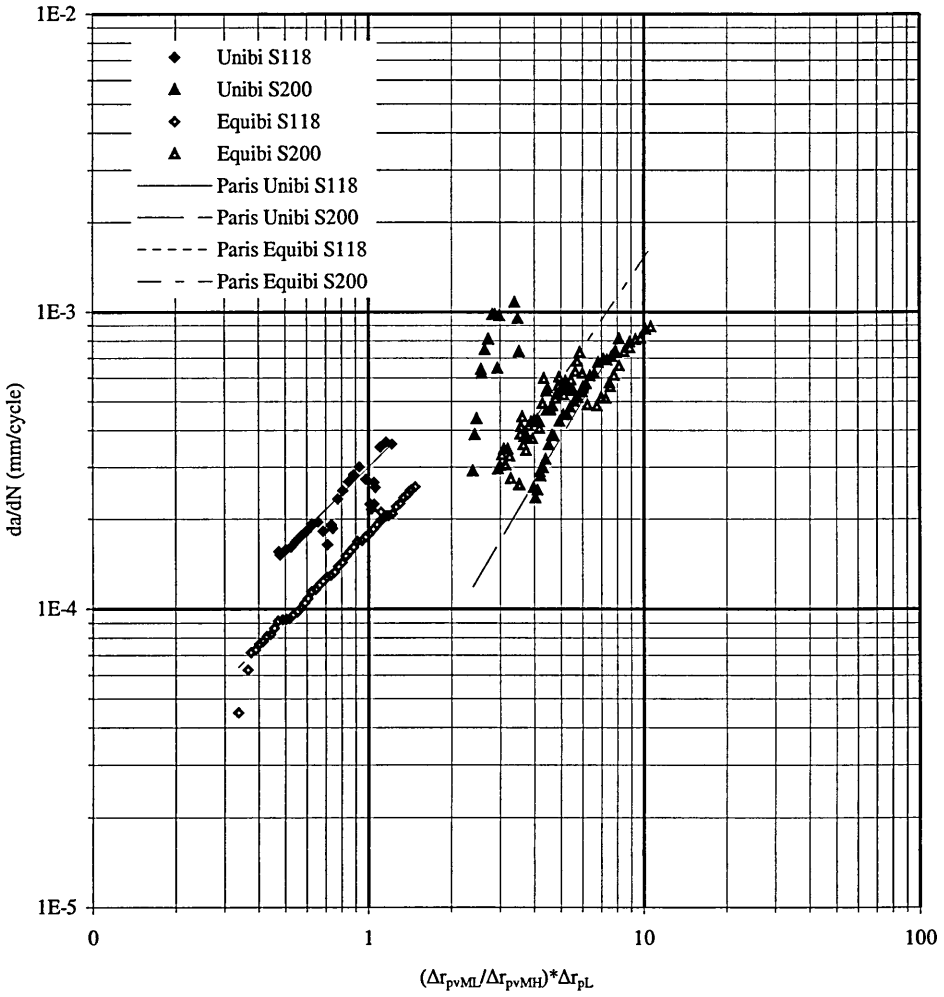


Figure 116: Crack growth rates for isothermal uniform thickness cruciform models using biaxial parameter with change in normal plastic zone size (experimental [124]/FE).

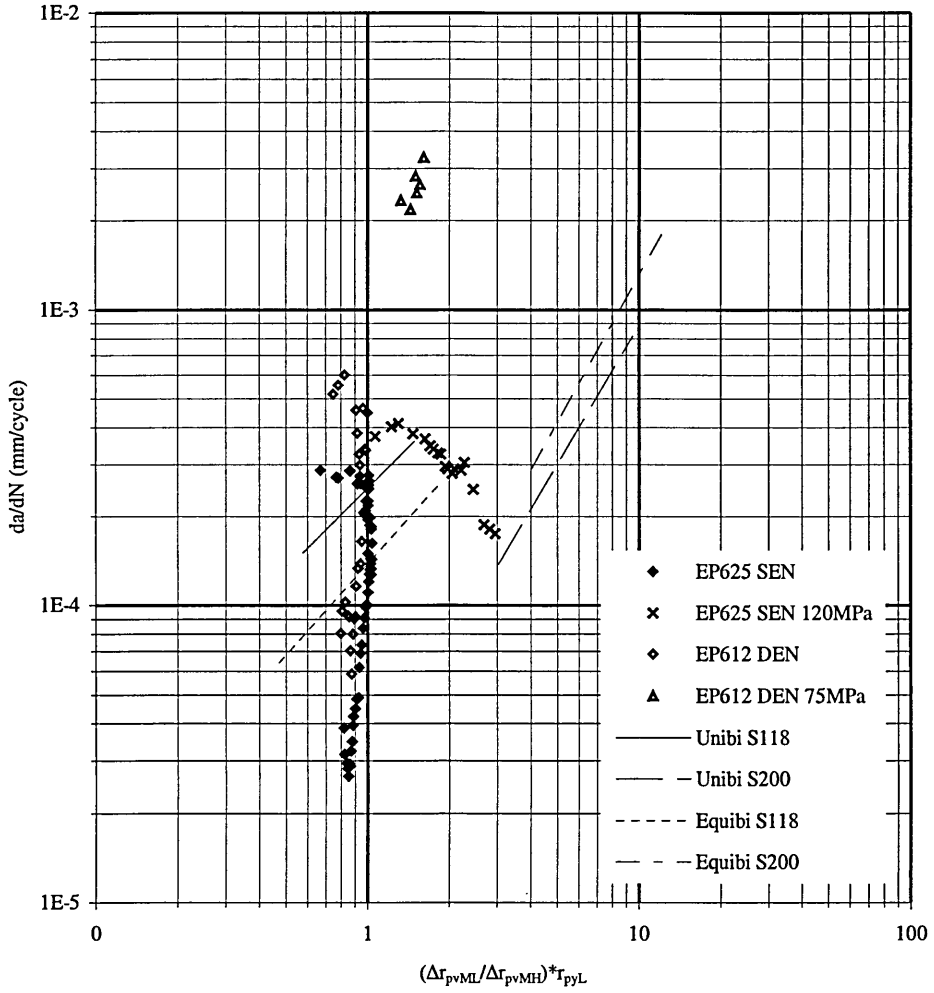


Figure 117: Correlation of thermal shock & isothermal uniform thickness cruciform crack growth rates with biaxial parameter using maximum load normal plastic zone size (experimental [124]/FE).

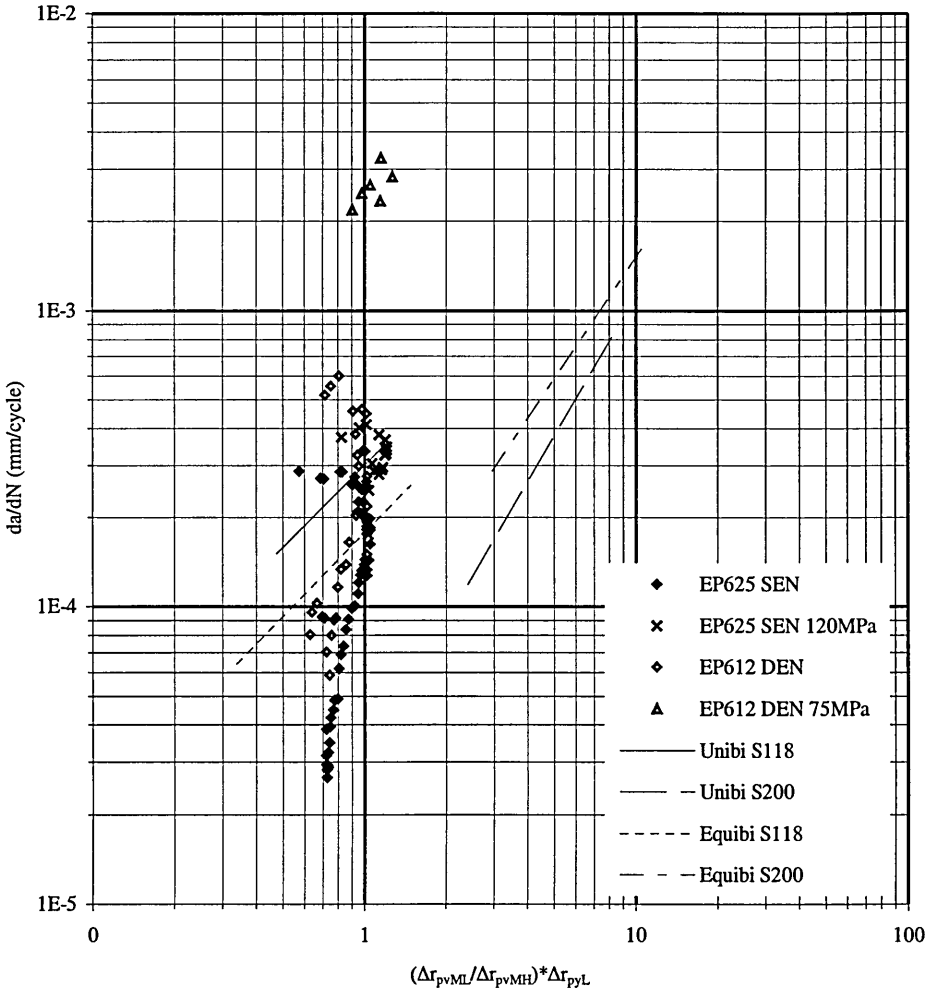


Figure 118: Correlation of thermal shock & isothermal uniform thickness cruciform crack growth rates with biaxial parameter using change in normal plastic zone size (experimental [124]/FE).

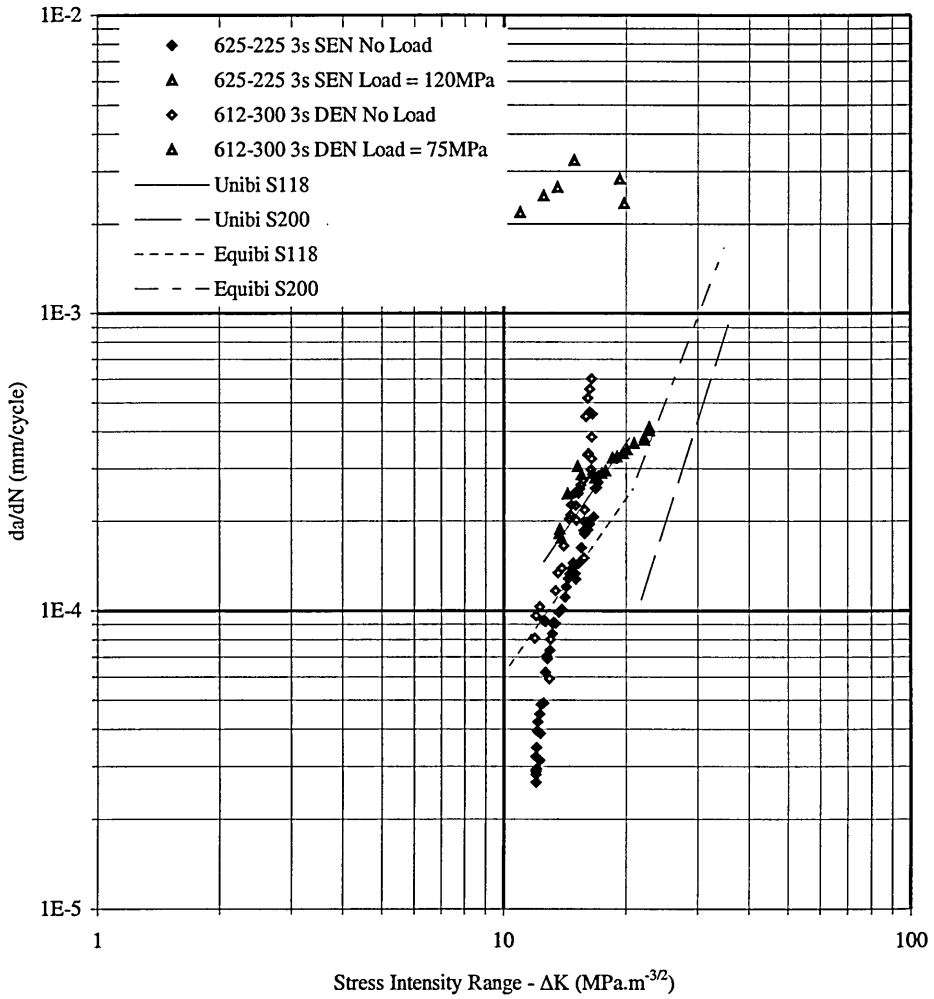


Figure 119: Correlation of elastic-plastic thermal shock & linear-elastic isothermal uniform thickness cruciform models with stress intensity factor (experimental [124]/FE).

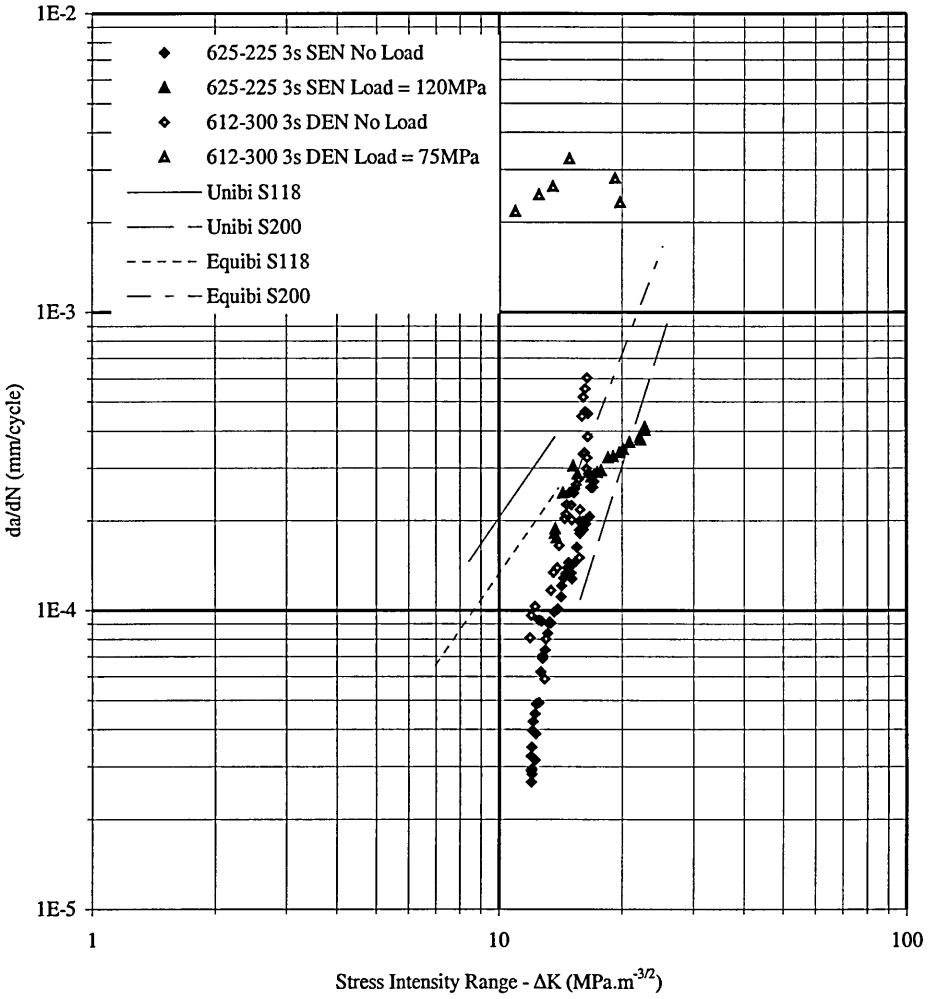


Figure 120: Correlation of elastic-plastic thermal shock & elastic-plastic isothermal uniform thickness cruciform models with stress intensity factor (experimental [124]/FE).

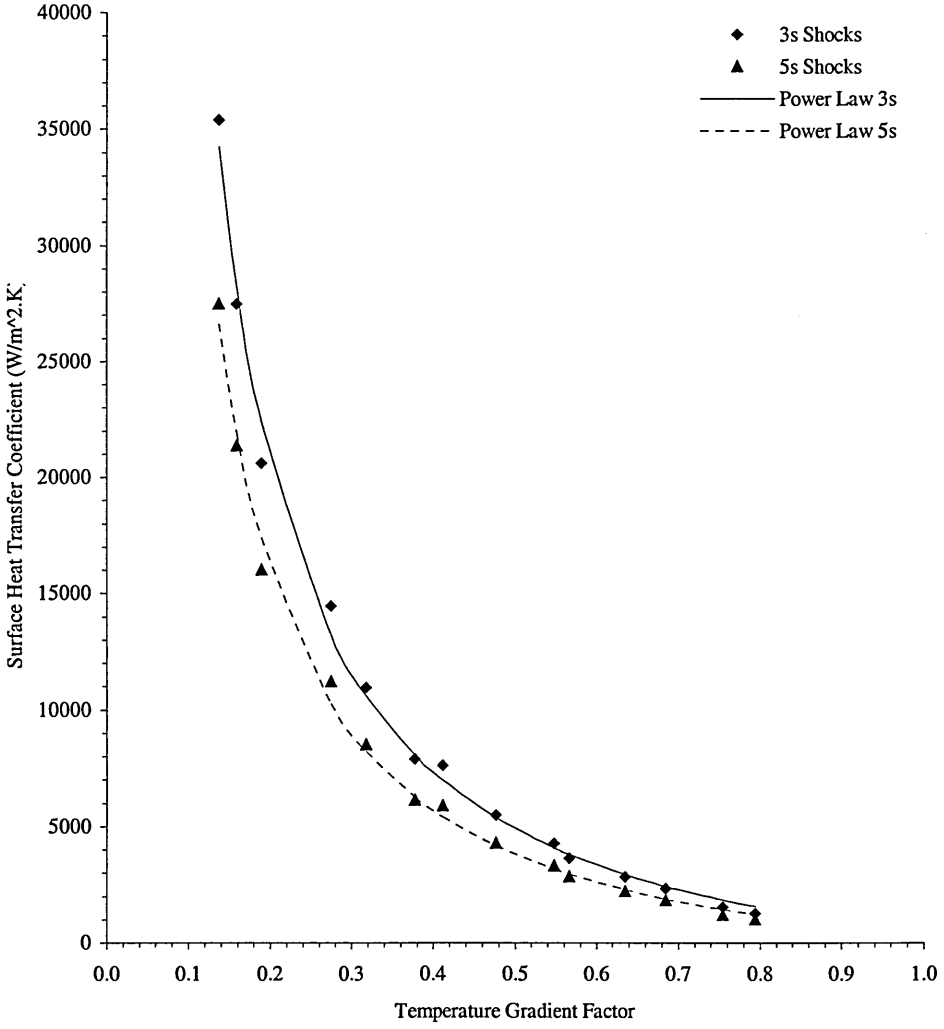


Figure 121: Power law relationship of surface heat transfer coefficients to an explicit thermal cycle definition.

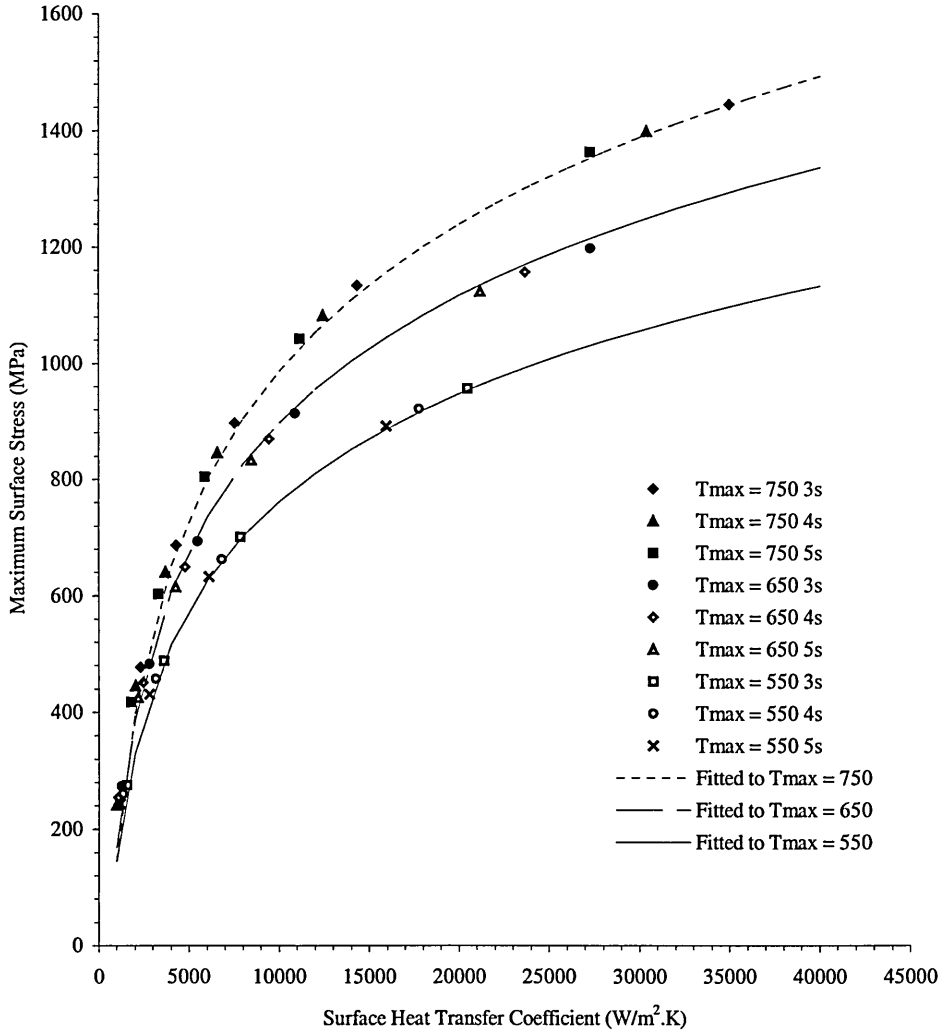


Figure 122: Maximum elastic stresses by FD&T.

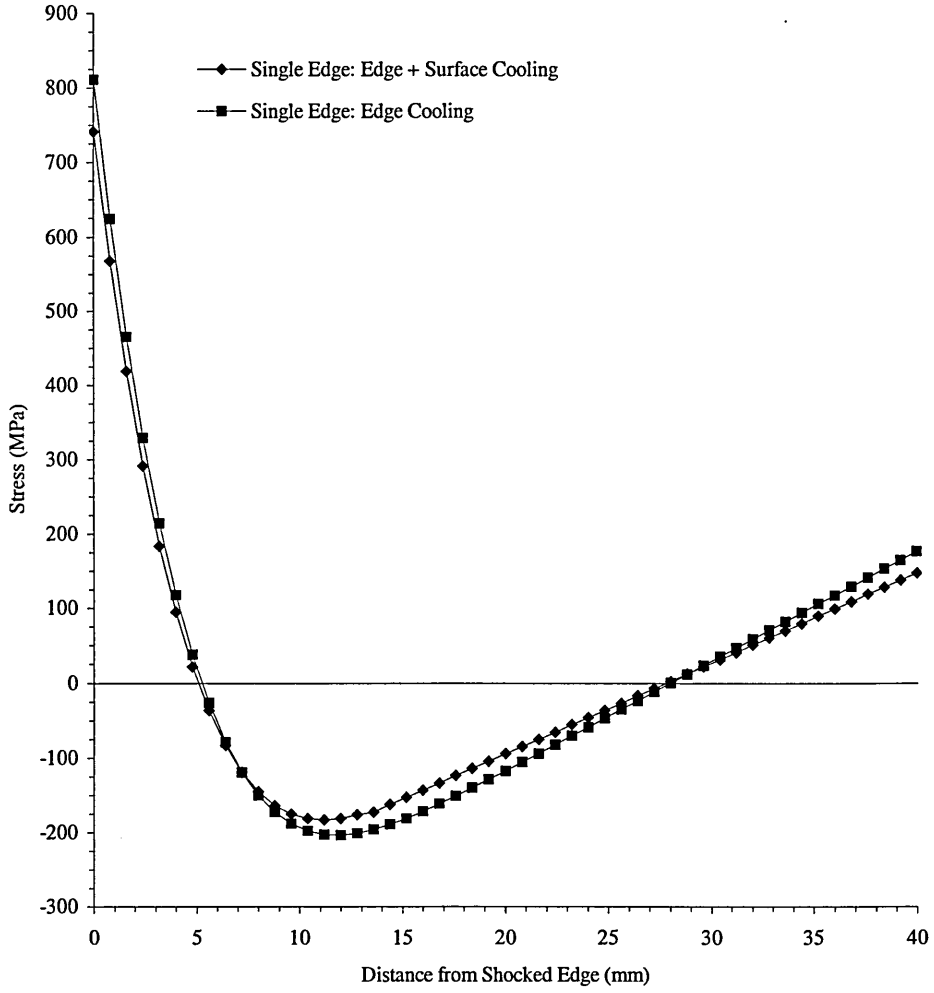


Figure 123: Single edge stresses for a 625-225°C in 3s shock for experimental [124] and model thermal FD&T distributions.

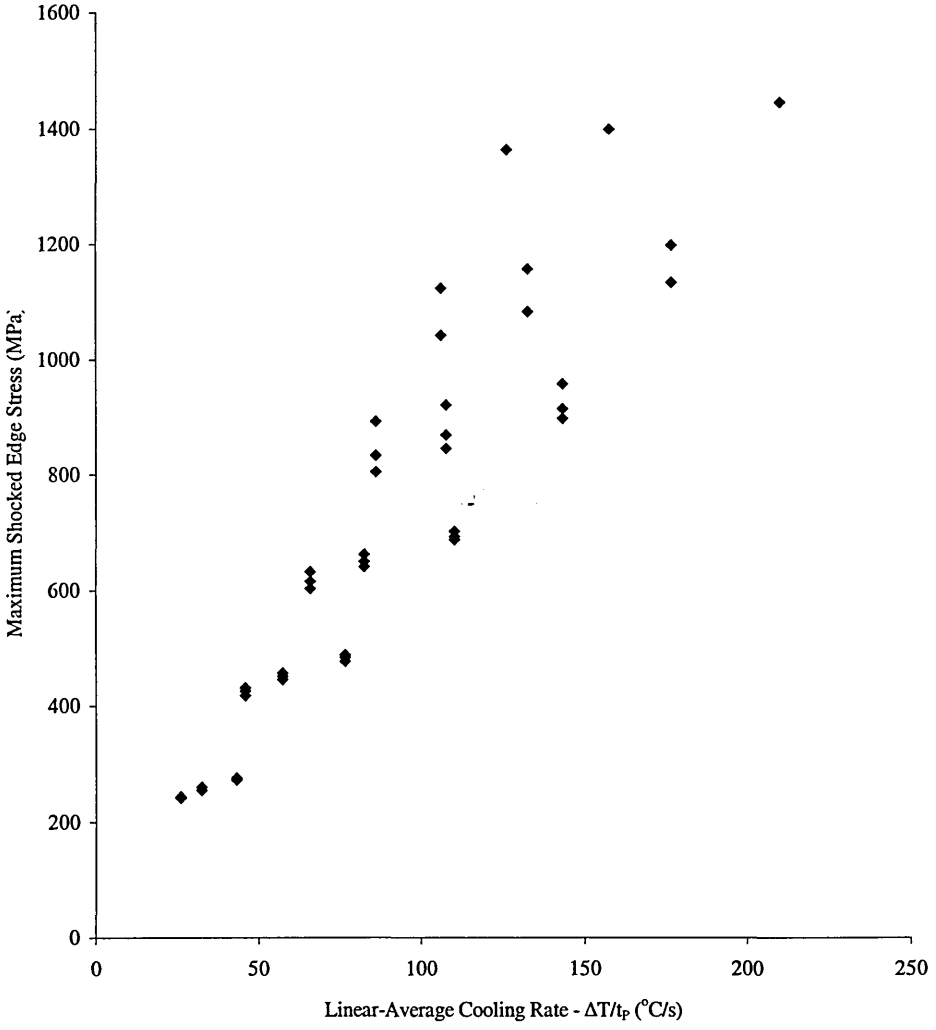


Figure 124: Linear-average cooling rate to maximum tensile stress.

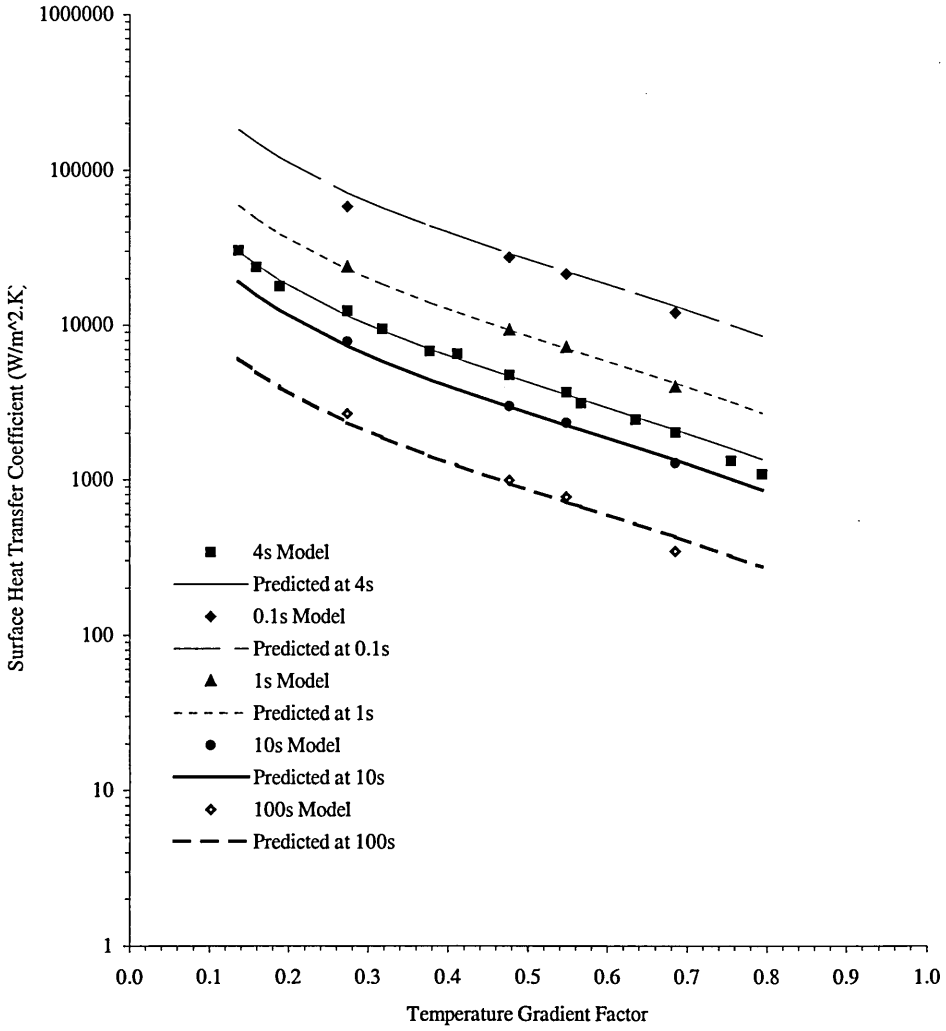


Figure 125: Predicted h values with eqn. 79.

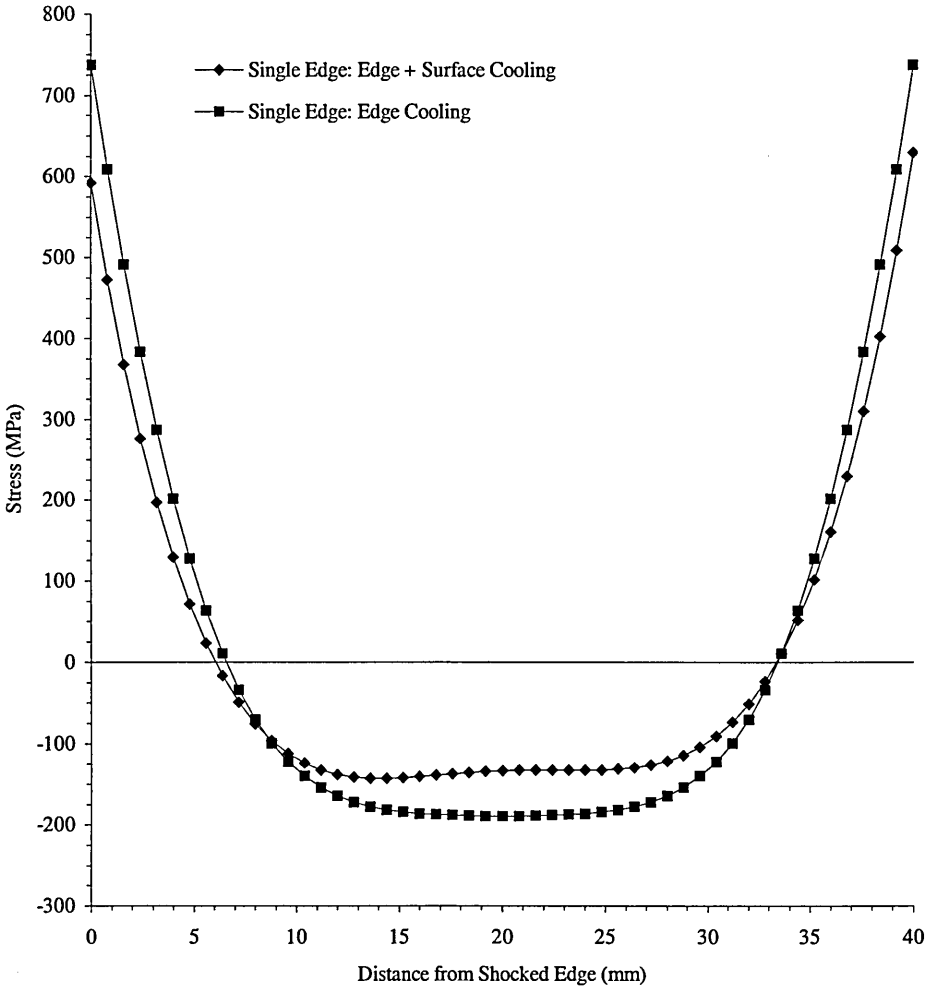


Figure 126: Double edge stresses for a 612-300°C in 3s shock for experimental [124] and model thermal FD&T distributions.

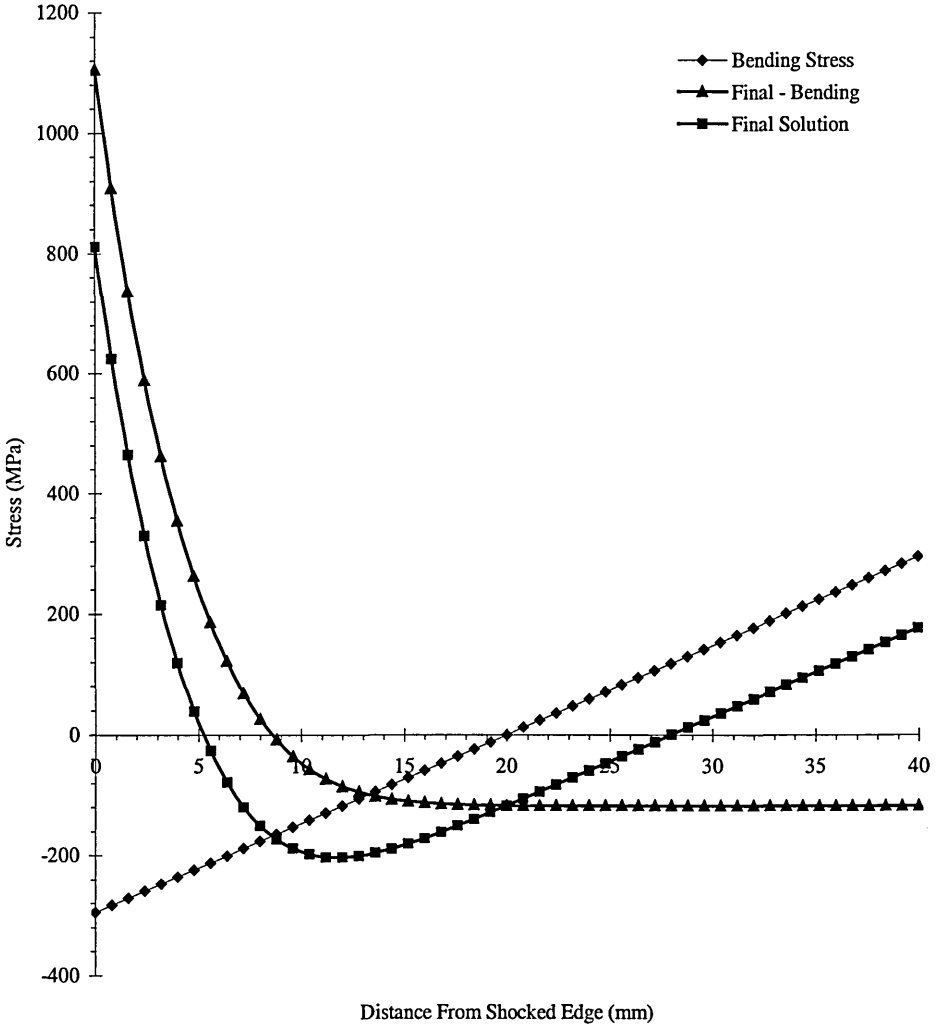


Figure 127: Summation of bending and non-linear membrane stresses for 625-2245°C in 3s shock.

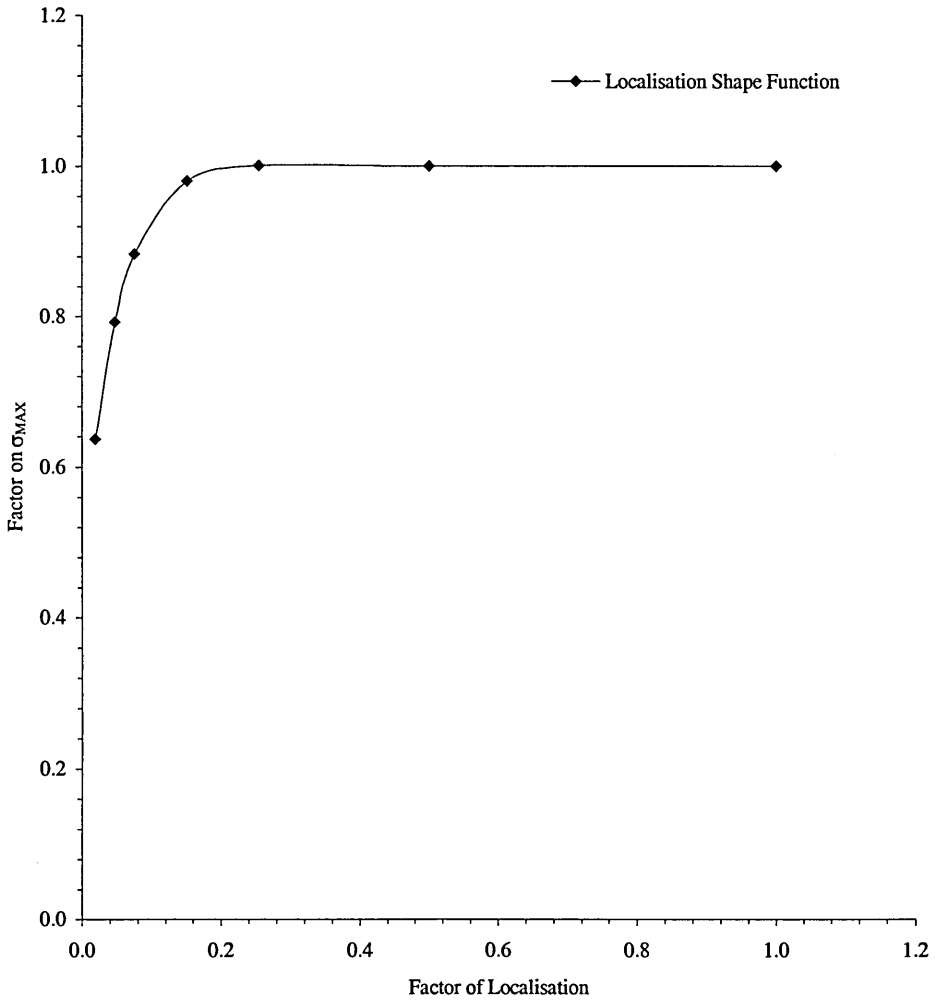


Figure 128: Normalisation of localised thermal shock effect.

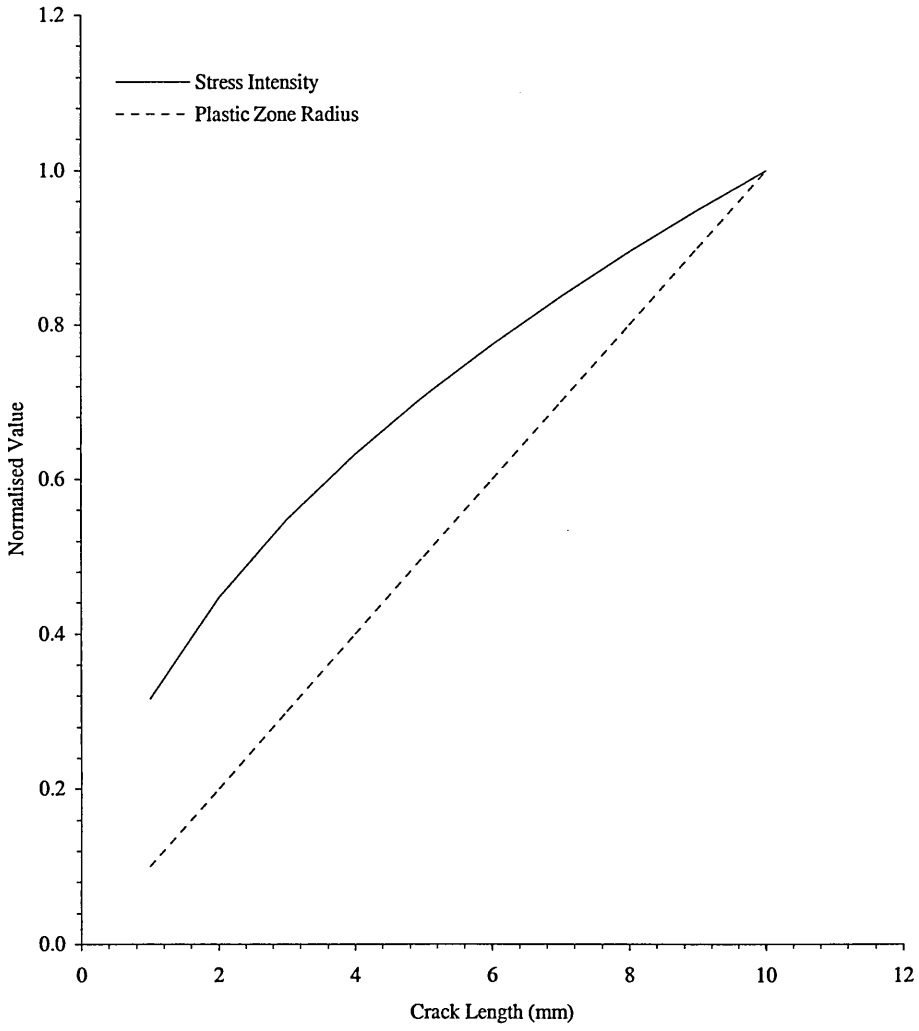


Figure 129: Normalised stress intensity factor and plastic zone size for increasing crack length and fixed applied load.

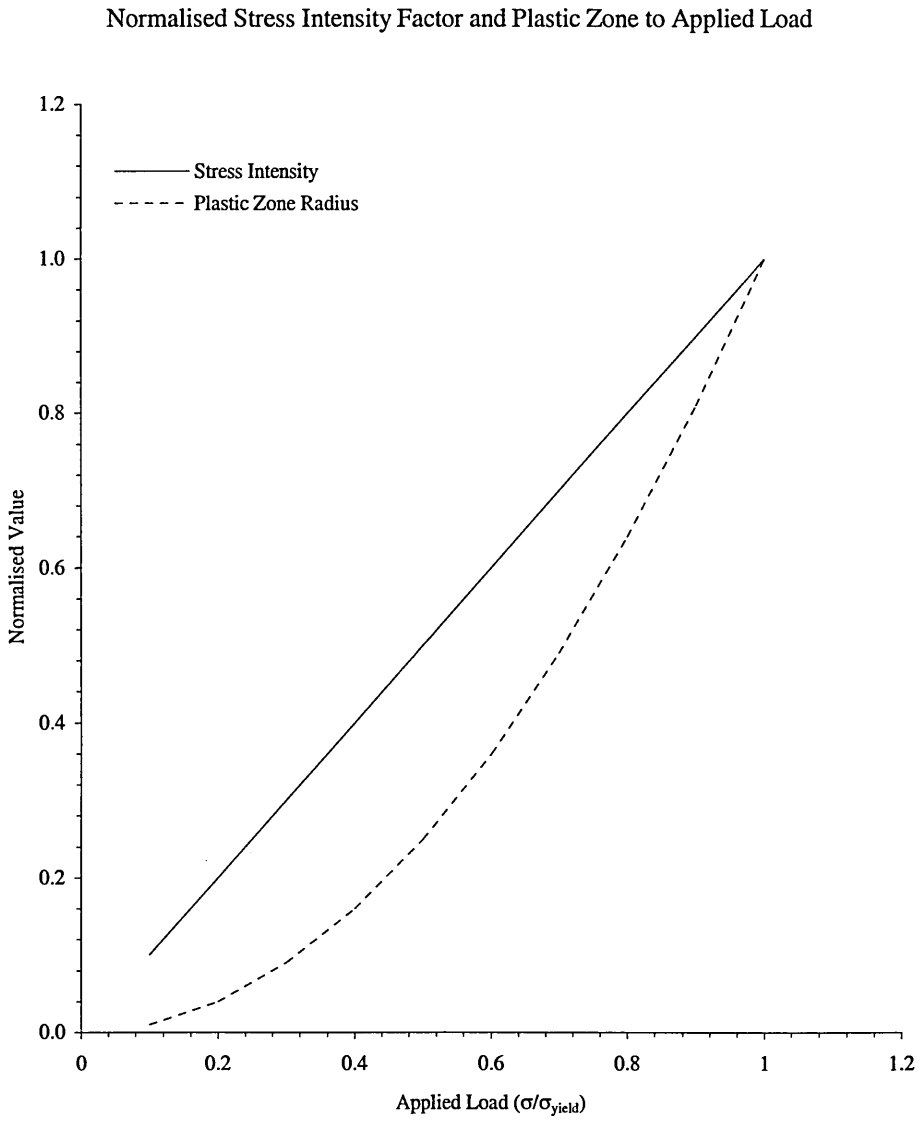


Figure 130: Normalised stress intensity factor and plastic zone size for increasing applied load and fixed crack length.

Appendix A

Inclusion of T-Stress into Theory of Elasticity Plastic Zone

Since T-Stress is an integral part of biaxial plastic zone sizes a relationship must be determined to describe its inclusion on the von Mises yield criterion of section 2.3. Following is a summary of the derivation to this effect.

The principal stresses as determined by Mohr's stress circle are as follows,

$$\sigma_{1,2} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\frac{\sigma_x - \sigma_y}{2} + \tau_{xy}}$$

$$\sigma_3 = \nu(\sigma_1 + \sigma_2)$$

eqn. 1

The von Mises yield criterion is written as follows,

$$\sigma_{yield}^2 = (\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2$$

eqn. 2

Using eqn. 18 and rewriting terms as below,

$$A = \frac{K_I}{\sqrt{2\pi r}} \qquad B = \frac{K_{II}}{\sqrt{2\pi r}} \qquad C = \cos\frac{\theta}{2} \qquad D = \sin\frac{\theta}{2}$$

$$E = \sin\frac{\theta}{2}\sin\frac{3\theta}{2} \qquad F = \cos\frac{\theta}{2}\cos\frac{3\theta}{2} \qquad T = \sigma(1-\Lambda)\cos 2\alpha$$

$$\sigma_x = AC[1-E] - BD[2+F] + T$$

$$\sigma_y = AC[1+E] - BDF$$

$$\sigma_x = ACF + BC[1-E]$$

eqn. 3

With Mohr's stress terms given as

$$VM_1 = \frac{\sigma_x + \sigma_y}{2} \quad \text{and} \quad VM_2 = \sqrt{\frac{\sigma_x - \sigma_y}{2} + \tau_{xy}}$$

eqn. 4

the von Mises yield criterion can be written in simpler form as

$$\sigma_{yield}^2 = VM_1^2 + 3VM_2^2 \qquad \text{for plane stress}$$

$$\sigma_{yield}^2 = VM_1^2(1-2\nu) + 3VM_2^2 \qquad \text{for plane strain}$$

eqn. 5

By substituting eqn. 3 into eqn. 4 and then into eqn. 5 we obtain the following

Plane stress

$$\begin{aligned}\sigma_{yield}^2 = {} & A^2\left[C^2 + 3C^2E^2 + 3D^2F^2\right] \\ & + B^2\left[4D^2 + 6D^2F + 3D^2F^2 + 3C^2 - 6C^2E + 3C^2E^2\right] \\ & + AB[6CDE + 6CDF - 2CD] \\ & + ACT[1-3E] - BDT[4+3F] + T^2\end{aligned}$$

eqn. 6

Plane strain

$$\begin{aligned}\sigma_{yield}^2 = {} & A^2\left[C^2(1-2\nu)^2 + 3C^2E^2 + 3D^2F^2\right] \\ & + B^2\left[D^2(1-2\nu)^2 + 3D^2 + 6D^2F + 3D^2F^2 + 3C^2 - 6C^2E + 3C^2E^2\right] \\ & + AB\left[6CDE + 6CDF - 2CD(1-2\nu)^2\right] \\ & + ACT\left[(1-2\nu)^2 - 3E\right] - BDT\left[(1-2\nu)^2 + 3F + 3\right] + T^2(1-\nu+\nu)^2\end{aligned}$$

eqn. 7

Using trigonometric identities this creates the same set of equations as those in section 2.3 with the addition of an extra term, here the last term in eqn. 6 and eqn. 7.

When written in full, the von Mises yield criterion is as follows,

Plane Stress

$$\sigma_{yield}^2 = \frac{K_I}{2\pi r}\cos^2\frac{\theta}{2}\left[3\sin^2\frac{\theta}{2}+1\right]+\frac{K_I K_{II}}{2\pi r}\sin\theta[3\cos-1]$$

$$+\frac{K_{II}}{2\pi r}\left[3+\sin^2\left(1-9\cos^2\frac{\theta}{2}\right)\right]$$

$$+\sigma(1-\Lambda)\cos 2\alpha\begin{bmatrix}\frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[1-3\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]\\ -\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left[4+3\cos\frac{\theta}{2}\cos\frac{3\theta}{2}\right]\\ +\sigma(1-\Lambda)\cos 2\alpha\end{bmatrix}$$

Plane Strain

$$\sigma_{yield}^2 = \frac{K_I}{2\pi r}\cos^2\frac{\theta}{2}\left[3\sin^2\frac{\theta}{2}+(1-2\nu)^2\right]+\frac{K_I K_{II}}{2\pi r}\sin\theta\left[3\cos-(1-2\nu)^2\right]$$

$$+\frac{K_{II}}{2\pi r}\left[3+\sin^2\left((1-2\nu)^2-9\cos^2\frac{\theta}{2}\right)\right]$$

$$+\sigma(1-\Lambda)\cos 2\alpha\begin{bmatrix}\frac{K_I}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left[(1-2\nu)^2-3\sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right]\\ -\frac{K_{II}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\left[(1-2\nu)^2+3\cos\frac{\theta}{2}\cos\frac{3\theta}{2}+3\right]\\ +\sigma(1-\Lambda)(1-\nu+\nu^2)\cos 2\alpha\end{bmatrix}$$

These equations must either be solved numerically for r or by rearranging to a quadratic that solves for $1/\sqrt{r}$ in the range of $\pi > \theta > -\pi$. The quadratic solution produces positive and negative solutions for different values of θ. It must however be remembered that the solution is for $1/\sqrt{r}$ and not r therefore it is not directly obvious which solution is the correct one, since to find r any negative sign will be illuminated by squaring. In order to confirm the solution, the two values obtained for $1/\sqrt{r}$ must be replaced in the original quadratic, this generates a single true solution, though at high loads the solution is unstable and the quadratic check fails in both cases.

Appendix B.1

Programme Code: Thermal Analysis

Temperature Dependant Finite Difference Solution for an Explicitly Described One-Dimensional Thermal Shock Transient Thermal Distribution

```
/* 2D FINITE DIFFERENCE
        TRANSIENT HEAT CONDUCTION
        TEMPERATURE DEPENDANCE

        Thin Rectangular Plate: Symmetry on bottom horizontal edge
                                Shocked on lower right edge,
                                height of this region is only half
                                of the full shock region due to symmetry */

/* LIBRARIES */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>

/* GLOBAL VARIABLES */
char  reply, go, prntoscrn;
char  sumname[11];
int   check,check1, tcnt, stop, switchto1,switchto2, iterend,iternum;
long  tosouth,tonorth, linelength, rcnt,ccnt, Nwnodes,Nhnodes,Nshnodes;
double initemp,endtemp,Ta;
double width,height,shock, dx,dy;
double Nwelements,Nhelements,Nshelements, eqnno;
double Kmax,Kn,Ks,Ke,Kw, densitymax,density;
double Cp,h,oldh;
double Elen, tperiod,minc,oldminc,tinc,oldtinc, Bi,Fo,oldFo;
double Fon,Fos,Foe,Fow, C;
double Nincrements,outinc;
double timer,timeout, sumcnt;
double Tp,Tptp1, Tn,Ts,Te,Tw;
double Tshock,Terror,Tfac;


/* GLOBAL ARRAYS */
double huge temps1[200][1000];
double huge temps2[200][1000];

/* FILES */
FILE *table;
FILE *fp;
FILE *ilog;


/* SUB-ROUTINES */
/* User definitions and inputs */
void intro();
void modelsettings();
void thermalproperties();
double inimaxtime();
double timeset(double minc);
int tperiodcheck();
double resultsoutput();
void numbereqns();

/* Initialise array to start temperatures */
void initarray(long rcnt, long ccnt, double initemp);

/* Scanning Routines */
double leftcorner(long rcnt,long ccnt, int switchto1,int switchto2);
double topside(long rcnt,long ccnt, int switchto1,int switchto2);
double rightcorner(long rcnt,long ccnt, int switchto1,int switchto2);
double leftside(long rcnt,long ccnt, int switchto1,int switchto2);
double internal(long rcnt,long ccnt, int switchto1,int switchto2);
double rightside(long rcnt,long ccnt, int switchto1,int switchto2);
double shocked(long rcnt,long ccnt, int switchto1,int switchto2);
double leftsymm(long rcnt,long ccnt, int switchto1,int switchto2);
```

```
double internalsymm(long rcnt,long ccnt, int switchto1,int switchto2);
double shockedsymm(long rcnt,long ccnt, int switchto1,int switchto2);

void summarypre(double timer, double sumcnt);
void itersummary();
void timerstart();
void timerend();

main()
{
 /* SHOW PROGRAMME INFORMATION */
 intro();

 /*REQUEST TEMPERATURE ENVIRONMENT */
 printf("\nINITIAL UNIFORM TEMPERATURE (C) = ");
 scanf(" %lf", &initemp);
 printf("FINAL SURFACE TEMPERATURE (C) = ");
 scanf(" %lf", &endtemp);
 printf("AMBIENT TEMPERATURE (C) = ");
 scanf(" %lf", &Ta);

 /* Calculate maximum temperature properties */
 Kmax = 15.0+(0.013*initemp);
 densitymax = 7900.0-(0.5*initemp);
 Kn = Kmax; Ks = Kmax; Ke = Kmax; Kw = Kmax;
 density = densitymax;

 /* GENERATE MODEL DATA, THERMAL PROPERTIES & TIME INCREMENT */
 check  = 0;
 check1 = 0;
 while (check != 1)
 {
  while (check1 != 1)
  {
            modelsettings();
            thermalproperties();
            minc = inimaxtime();

            /* Provide oppurtunity to reset all model and thermal properties */
            printf("\nMAXIMUM TIME INCREMENT IS ** %f seconds **\n", minc);
            printf("IS THIS INCREMENT OF AN ACCEPTABLE ORDER? Y/N: ");
            scanf(" %c", &reply);


            if ((reply == 'Y') || (reply == 'y'))
            {
             /* Provide time period */
             printf("\nREQUIRED TIME PERIOD (s) = ");
             scanf(" %lf", &tperiod);
             tinc = timeset(minc);
             /* Calculate the new Fourier number to be applied */
             Fo = (Kmax*tinc)/(densitymax*Cp*Elen*Elen);
             printf("\nMAXIMUM Fourier NUMBER is: %lf\n", Fo);
             printf("MAXIMUM TIME INCREMENT IS: %lf\n", minc);
             printf("ACTUAL TIME INCREMENT USED IS: %lf\n", tinc);
            }
 }
 /* Ensure time period is greater than time increment */
 check = tperiodcheck();
 }

 /* DEFINE NUMBER OF NODES */
 Nwnodes  = (long)Nwelements + 1L;
 Nhnodes  = (long)Nhelements + 1L;
 Nshnodes = (long)Nshelements + 1L;

 /* SET-UP OUTPUT INTERVAL FOR RESULTS */
```

```
outinc = resultsoutput();

/* SHOW NUMBER OF CALCULATIONS TO ABORT OR CONTINUE */
numbereqns();
printf("\nPrint analysis summary to screen? [Y/N]");
scanf(" %c", &prntoscrn);
printf("\n\n");

if ((fp = fopen("array.txt", "w+")) == NULL)
{
 printf("\n\n*** Cannot open file array.txt ***\n\n");
 exit(0);
}

/* CALCULATE FINITE DIFFERENCE SOLUTION */
/* Open iteration summary file "2DLOG.txt" */
if ((ilog = fopen("2Dlog.txt", "w+")) == NULL)
{
 printf("\n\n*** Cannot open file 2Dlog.txt ***\n\n");
 exit(0);
}
timerstart();

iternum = 1;
iterend = 0;
while (iterend == 0)
{
 switchto1 = 1;
 switchto2 = 0;
 timer = tinc;
 timeout = tinc;
 sumcnt = 1;

 /* Re-initialise the temperature array */
 rcnt = 0;
 ccnt = 0;
 initarray(rcnt,ccnt, initemp);

 while (timer <= tperiod+(tinc/2))
 {
          rcnt = 0;
          while (rcnt <= Nhelements)
          {
           ccnt = 0;
           while (ccnt <= Nwelements)
           {
           /* Scan appropriate values */
           if ( (rcnt == 0) && (ccnt == 0) )
           { leftcorner(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt == 0) && (ccnt > 0) && (ccnt < Nwelements) )
           { topside(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt == 0) && (ccnt == Nwelements) )
           { rightcorner(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt > 0) && (rcnt < Nhelements)  && (ccnt == 0) )
           { leftside(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt > 0) && (rcnt < Nhelements) && (ccnt > 0) && (ccnt < Nwelements) )
           { internal(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt > 0) && (rcnt <= (Nhelements-Nshnodes)) && (ccnt == Nwelements) )
           { rightside(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt > (Nhelements-Nshnodes)) && (rcnt < Nhelements) && (ccnt == Nwelements) )
           { shocked(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt == Nhelements) && (ccnt == 0) )
           { leftsymm(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt == Nhelements) && (ccnt > 0) && (ccnt < Nwelements) )
           { internalsymm(rcnt,ccnt, switchto1,switchto2); }
           if ( (rcnt == Nhelements) && (ccnt == Nwelements) )
           { shockedsymm(rcnt,ccnt, switchto1,switchto2); }
```

```
            ccnt ++;
            }
            rcnt ++;
            }
            timer = timer + tinc;
            timeout = timeout + tinc;
            /* Set swap switches to alternative array
                        Switchto1 for array1 & Switchto2 for array2 */
            if ((switchto1 == 1) && (switchto2 == 0))
                        { switchto1 = 0; switchto2 = 1; }
            else
                        { switchto1 = 1; switchto2 = 0; }
 }

 if ((switchto1 == 1) && (switchto2 == 0))
 { Tshock = temps1[Nhnodes][Nwnodes]; }
 if ((switchto1 == 0) && (switchto2 == 1))
 { Tshock = temps2[Nhnodes][Nwnodes]; }

 Terror = Tshock - endtemp;
 Tfac = Terror/endtemp;

 oldh = h;
 h = h + (h*Tfac);
 oldminc = minc;
 minc = inimaxtime();
 oldtinc = tinc;
 tinc = timeset(minc);

 if ( (Tfac > -0.0009) && (Tfac < 0.0009) )
 { iterend = 1; }

 itersummary();
 iternum ++;
 }

 printf("\nSurface Heat Transfer   = %lf\n", h);
 printf("Maximum Time Increment  = %lf\n", minc);
 printf("Time Increment used     = %lf\n", tinc);

 timerend();
 printf("\n\nDONE");
 fclose(fp);
 fclose(ilog);
 return (0);
}


/* PROGRAMME INFORMATION */
void intro()
{
 printf("\n\n***********************************************\n");
 printf("**Programme: 2dconv.c\n");
 printf("**Pure convection FINITE DIFFERENCE programme\n");
 printf("  To calculate the required\n");
 printf("  'surface heat transfer coefficient (h)'");
 printf("\n***********************************************\n");

 return;
}

/* MODEL DATA: DIMENSIONS & ELEMENT QUANTITY */
void modelsettings()
{
 printf("\n**MODEL SETTINGS: WIDTH & HEIGHT\n");
 printf("Width (mm) = ");
```

```
 scanf(" %lf", &width);
 width = width/1000;
 printf("Height (mm) = ");
 scanf(" %lf", &height);
 height = height/1000;
 printf("Height of the shocked region (mm) = ");
 scanf(" %lf", &shock);

 shock = shock/1000;
 printf("\nNUMBER OF ELEMENTS ACROSS THE WIDTH = ");
 scanf(" %lf", &Nwelements);
 dx = width/Nwelements;
 Nhelements = height/dx;
 Nhelements = floor(Nhelements+0.5);
 dy = height/Nhelements;
 printf("SQUARE ELEMENTS MAINTAINED: %.0lf ELEMENTS THROUGH THE HEIGHT", Nhelements);

 Nshelements = (shock/height)*Nhelements;

 Nshelements = ceil(Nshelements);

 printf("\nNUMBER OF SHOCKED ELEMENTS IS %.0lf", Nshelements);

 return;
}

/* THERMAL PROPERTY DEFINITION */
void thermalproperties()
{
 printf("\n\n**SPECIFY THERMAL PROPERTIES\n");
 printf("SPECIFIC HEAT (J/KgK)= ");
 scanf(" %lf", &Cp);
 printf("SUGGESTED SURFACE HEAT TRANSFER COEFFICIENT (W/m2K)= ");
 scanf(" %lf", &h);

 return;
}

/* INITIAL MAXIMUM TIME INCREMENT */
double inimaxtime()
{
 /* Calculate maximum allowable time increment
            for inital surface heat transfer coefficient */
 if (dy <= dx) { Elen = dy; }
 if (dx <= dy) { Elen = dx; }
 Bi = (4*Kmax)/(2*Elen*h);
 minc = (Elen*densitymax*Cp)/(2*h*(Bi+1));


 return(minc);
}

/* TIME INCREMENT SELECTION */
double timeset(double minc)
{
 int tcheck, order;

 check1 = 1;
 tcheck = 0;
 order = 1;
 while (tcheck == 0)
 {
  if ((minc >= 0.50/order) && (minc < tperiod))
  {
            tinc = 0.50/order;
            tcheck = 1;
  }
```

```
 if ((minc >= 0.25/order) && (minc < 0.50/order))
 {
        tinc = 0.25/order;
        tcheck = 1;
 }
 if ((minc >= 0.20/order) && (minc < 0.25/order))
 {
        tinc = 0.20/order;
        tcheck = 1;
 }
 if ((minc >= 0.10/order) && (minc < 0.20/order))
 {
        tinc = 0.10/order;
        tcheck = 1;
 }
 order = order*10;
 }

 return(tinc);
}

int tperiodcheck()
{
 while (tperiod <= tinc)
 {
  printf("\n***ERROR: The selected time increment, %.4g, is greater than", tinc);
  printf("\n       or equal to the overall time period, %.4g", tperiod);
  printf("\n\nRE-SPECIFY THE REQUIRED TIME PERIOD: ");
  scanf(" %lf", &tperiod);
 }

 if (tperiod > tinc) { check = 1; }

 return (check);
}

void numbereqns()
{
 char stop;

 eqnno = Nwnodes*Nhnodes*tperiod/tinc;
 printf("\nThere will be %g calculations to conduct/iteration\n", eqnno);
 printf("Do you wish to continue? [Y/N]: ");
 scanf(" %c", &stop);
 if ((stop != 'Y') && (stop != 'y'))
 {
  if (eqnno >= 1e6)
  {
        printf("\n!!*!?Holy Panicking Processors Batman!!*?!");
        printf("\n**!??Let's Get the Heck Out of Here!**?!\n");
  }
  printf("\nProgramme Terminated");
  exit(0);
 }


 return;
}
/* SET INTERVAL FOR OUTPUT OF RESULTS */
double resultsoutput()
{
 Nincrements = tperiod/tinc;
 printf("\nTHERE WILL BE %g INCREMENTS", Nincrements);
 printf("\nAT WHAT TIME INTERVAL DO YOU WISH TO OUTPUT VALUES: ");
 scanf(" %lf", &outinc);

 while (outinc > tperiod)
```

```
{
 printf("\n ***ERROR: This time interval must be smaller");
 printf("\n          than the time period %.1f", tperiod);
 printf("\n\nRE-SPECIFY THE OUTPUT TIME INTERVAL: ");
 scanf(" %lf", &outinc);
 }

 if (tperiod/outinc > 5.0 )
 {
 printf("\n***WARNING THIS WILL GENERATE %.0lf SUMMARY FILES PER ITERATION",
                                tperiod/outinc);
 printf("\n   INTERVAL TO CREATE 5 SUMMARY FILES IS %lf", tperiod/5.0);
 printf("\n   CONFIRM THE INTERVAL: ");
 scanf(" %lf", &outinc);
 }

 return(outinc);
}

/* INITIALISE ARRAY TO START TEMPERATURES */
void initarray(long rcnt, long ccnt, double initemp)
{
 rcnt = 1;
 while (rcnt <= Nhnodes)
 {
  ccnt = 1;
  while (ccnt <= Nwnodes)
  {
            if ((switchto1 == 1) && (switchto2 == 0))
            { temps1[rcnt][ccnt] = initemp; }
            if ((switchto1 == 0) && (switchto2 == 1))
            { temps2[rcnt][ccnt] = initemp; }

            ccnt ++;
  }
  rcnt ++;
 }

 return;
}


/* SCAN APPROPRIATE VALUES FOR CALCULATION */
double leftcorner(long rcnt,long ccnt, int switchto1,int switchto2)
{
 /* scan appropriate array */
 if ((switchto1 == 1) && (switchto2 == 0))
 {
  Tp = temps1[rcnt+1][ccnt+1];
  Te = temps1[rcnt+1][ccnt+2];
  Ts = temps1[rcnt+2][ccnt+1];
 }
 if ((switchto1 == 0) && (switchto2 == 1))
 {
  Tp = temps2[rcnt+1][ccnt+1];
  Te = temps2[rcnt+1][ccnt+2];
  Ts = temps2[rcnt+2][ccnt+1];
 }

 Ke = 15.0 + (0.013*Te);
 Ks = 15.0 + (0.013*Ts);
 density = 7900.0 - (0.5*Tp);

 Foe = (Ke*tinc)/(Elen*Elen*density*Cp);
 Fos = (Ks*tinc)/(Elen*Elen*density*Cp);

 Tptp1 = (2*Foe*Te)+(2*Fos*Ts)+(Tp*(1-(2*Foe)-(2*Fos)));
```

```
/* Write new value to alternative array */
if ((switchto1 == 1) && (switchto2 == 0))
{ temps2[rcnt+1][ccnt+1] = Tptp1; }
if ((switchto1 == 0) && (switchto2 == 1))
{ temps1[rcnt+1][ccnt+1] = Tptp1; }

if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
{
 summarypre(timer, sumcnt);
 fprintf(table, "%4ld %6.3lf  %7.3lf", rcnt+1,(height-(rcnt*(height/Nhelements)))*1000, Tptp1);
 sumcnt ++;
}

return(Tptp1);
}
double topside(long rcnt,long ccnt, int switchto1,int switchto2)
{
/* scan appropriate array */
if ((switchto1 == 1) && (switchto2 == 0))
{
 Tw = temps1[rcnt+1][ccnt];
 Tp = temps1[rcnt+1][ccnt+1];
 Te = temps1[rcnt+1][ccnt+2];
 Ts = temps1[rcnt+2][ccnt+1];
}
if ((switchto1 == 0) && (switchto2 == 1))
{
 Tw = temps2[rcnt+1][ccnt];
 Tp = temps2[rcnt+1][ccnt+1];
 Te = temps2[rcnt+1][ccnt+2];
 Ts = temps2[rcnt+2][ccnt+1];
}

Kw = 15.0 + (0.013*Tw);
Ke = 15.0 + (0.013*Te);
Ks = 15.0 + (0.013*Ts);
density = 7900.0 - (0.5*Tp);

Fow = (Kw*tinc)/(Elen*Elen*density*Cp);
Foe = (Ke*tinc)/(Elen*Elen*density*Cp);
Fos = (Ks*tinc)/(Elen*Elen*density*Cp);

Tptp1 = (Fow*Tw)+(Foe*Te)+(2*Fos*Ts)+(Tp*(1-Fow-Foe-(2*Fos)));
/* Write new value to alternative array */
if ((switchto1 == 1) && (switchto2 == 0))
{ temps2[rcnt+1][ccnt+1] = Tptp1; }
if ((switchto1 == 0) && (switchto2 == 1))
{ temps1[rcnt+1][ccnt+1] = Tptp1; }

if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
{ fprintf(table, "%9.3lf", Tptp1); }

return(Tptp1);
}
double rightcorner(long rcnt,long ccnt, int switchto1,int switchto2)
{
/* scan appropriate array */
if ((switchto1 == 1) && (switchto2 == 0))
{
 Tw = temps1[rcnt+1][ccnt];
 Tp = temps1[rcnt+1][ccnt+1];
 Ts = temps1[rcnt+2][ccnt+1];
}
if ((switchto1 == 0) && (switchto2 == 1))
{
 Tw = temps2[rcnt+1][ccnt];
 Tp = temps2[rcnt+1][ccnt+1];
```

```
 Ts = temps2[rcnt+2][ccnt+1];
 }

 Kw = 15.0 + (0.013*Tw);
 Ks = 15.0 + (0.013*Ts);
 density = 7900.0 - (0.5*Tp);

 Fow = (Kw*tinc)/(Elen*Elen*density*Cp);
 Fos = (Ks*tinc)/(Elen*Elen*density*Cp);

 Tptp1 = (2*Fow*Tw)+(2*Fos*Ts)+(Tp*(1-(2*Fow)-(2*Fos)));
 /* Write new value to alternative array */
 if ((switchto1 == 1) && (switchto2 == 0))
 { temps2[rcnt+1][ccnt+1] = Tptp1; }
 if ((switchto1 == 0) && (switchto2 == 1))
 { temps1[rcnt+1][ccnt+1] = Tptp1; }

 if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
 { fprintf(table, "%9.3lf\n", Tptp1); }

 return (Tptp1);
}
double leftside(long rcnt,long ccnt, int switchto1,int switchto2)
{
 /* scan appropriate array */
 if ((switchto1 == 1) && (switchto2 == 0))
 {
  Tn = temps1[rcnt][ccnt+1];
  Tp = temps1[rcnt+1][ccnt+1];
  Te = temps1[rcnt+1][ccnt+2];
  Ts = temps1[rcnt+2][ccnt+1];
 }
 if ((switchto1 == 0) && (switchto2 == 1))
 {
  Tn = temps2[rcnt][ccnt+1];
  Tp = temps2[rcnt+1][ccnt+1];
  Te = temps2[rcnt+1][ccnt+2];
  Ts = temps2[rcnt+2][ccnt+1];
 }

 Kn = 15.0 + (0.013*Tn);
 Ke = 15.0 + (0.013*Te);
 Ks = 15.0 + (0.013*Ts);
 density = 7900.0 - (0.5*Tp);

 Fon = (Kn*tinc)/(Elen*Elen*density*Cp);
 Foe = (Ke*tinc)/(Elen*Elen*density*Cp);
 Fos = (Ks*tinc)/(Elen*Elen*density*Cp);

 Tptp1 = (Fon*Tn)+(Fos*Ts)+(2*Foe*Te)+(Tp*(1-Fon-Fos-(2*Foe)));
 /* Write new value to alternative array */
 if ((switchto1 == 1) && (switchto2 == 0))
 { temps2[rcnt+1][ccnt+1] = Tptp1; }
 if ((switchto1 == 0) && (switchto2 == 1))
 { temps1[rcnt+1][ccnt+1] = Tptp1; }

 if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
 {
  fprintf(table, "%4ld %6.3lf  %7.3lf", rcnt+1,(height-(rcnt*(height/Nhelements)))*1000, Tptp1);
 }

 return(Tptp1);
}
double internal(long rcnt,long ccnt, int switchto1,int switchto2)
{
 /* scan appropriate array */
 if ((switchto1 == 1) && (switchto2 == 0))
```

```
{
 Tn = temps1[rcnt][ccnt+1];
 Tw = temps1[rcnt+1][ccnt];
 Tp = temps1[rcnt+1][ccnt+1];
 Te = temps1[rcnt+1][ccnt+2];
 Ts = temps1[rcnt+2][ccnt+1];
 }
 if ((switchto1 == 0) && (switchto2 == 1))
 {
 Tn = temps2[rcnt][ccnt+1];
 Tw = temps2[rcnt+1][ccnt];
 Tp = temps2[rcnt+1][ccnt+1];
 Te = temps2[rcnt+1][ccnt+2];
 Ts = temps2[rcnt+2][ccnt+1];
 }

 Kn = 15.0 + (0.013*Tn);
 Kw = 15.0 + (0.013*Tw);
 Ke = 15.0 + (0.013*Te);
 Ks = 15.0 + (0.013*Ts);
 density = 7900.0 - (0.5*Tp);

 Fon = (Kn*tinc)/(Elen*Elen*density*Cp);
 Fow = (Kw*tinc)/(Elen*Elen*density*Cp);
 Foe = (Ke*tinc)/(Elen*Elen*density*Cp);
 Fos = (Ks*tinc)/(Elen*Elen*density*Cp);

 Tptp1 = (Fon*Tn)+(Fos*Ts)+(Foe*Te)+(Fow*Tw)+(Tp*(1-Fon-Fos-Foe-Fow));
 /* Write new value to alternative array */
 if ((switchto1 == 1) && (switchto2 == 0))
 { temps2[rcnt+1][ccnt+1] = Tptp1; }
 if ((switchto1 == 0) && (switchto2 == 1))
 { temps1[rcnt+1][ccnt+1] = Tptp1; }

 if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
 { fprintf(table, "%9.3lf", Tptp1); }

 return(Tptp1);
}
double rightside(long rcnt,long ccnt, int switchto1,int switchto2)
{
 /* scan appropriate array */
 if ((switchto1 == 1) && (switchto2 == 0))
 {
 Tn = temps1[rcnt][ccnt+1];
 Tw = temps1[rcnt+1][ccnt];
 Tp = temps1[rcnt+1][ccnt+1];
 Ts = temps1[rcnt+2][ccnt+1];
 }
 if ((switchto1 == 0) && (switchto2 == 1))
 {
 Tn = temps2[rcnt][ccnt+1];
 Tw = temps2[rcnt+1][ccnt];
 Tp = temps2[rcnt+1][ccnt+1];
 Ts = temps2[rcnt+2][ccnt+1];
 }

 Kn = 15.0 + (0.013*Tn);
 Kw = 15.0 + (0.013*Tw);
 Ks = 15.0 + (0.013*Ts);
 density = 7900.0 - (0.5*Tp);

 Fon = (Kn*tinc)/(Elen*Elen*density*Cp);
 Fow = (Kw*tinc)/(Elen*Elen*density*Cp);
 Fos = (Ks*tinc)/(Elen*Elen*density*Cp);

 Tptp1 = (Fon*Tn)+(Fos*Ts)+(2*Fow*Tw)+(Tp*(1-Fon-Fos-(2*Fow)));
```

```
/* Write new value to alternative array */
if ((switchto1 == 1) && (switchto2 == 0))
{ temps2[rcnt+1][ccnt+1] = Tptp1; }
if ((switchto1 == 0) && (switchto2 == 1))
{ temps1[rcnt+1][ccnt+1] = Tptp1; }

if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
{ fprintf(table, "%9.3lf\n", Tptp1); }

return(Tptp1);
}
double shocked(long rcnt,long ccnt, int switchto1,int switchto2)
{
/* scan appropriate array */
if ((switchto1 == 1) && (switchto2 == 0))
{
 Tn = temps1[rcnt][ccnt+1];
 Tw = temps1[rcnt+1][ccnt];
 Tp = temps1[rcnt+1][ccnt+1];
 Ts = temps1[rcnt+2][ccnt+1];
}
if ((switchto1 == 0) && (switchto2 == 1))
{
 Tn = temps2[rcnt][ccnt+1];
 Tw = temps2[rcnt+1][ccnt];
 Tp = temps2[rcnt+1][ccnt+1];
 Ts = temps2[rcnt+2][ccnt+1];
}

Kn = 15.0 + (0.013*Tn);
Kw = 15.0 + (0.013*Tw);
Ks = 15.0 + (0.013*Ts);
density = 7900.0 - (0.5*Tp);

Fon = (Kn*tinc)/(Elen*Elen*density*Cp);
Fow = (Kw*tinc)/(Elen*Elen*density*Cp);
Fos = (Ks*tinc)/(Elen*Elen*density*Cp);
C = (h*tinc)/(Elen*density*Cp);

Tptp1 = (Fon*Tn)+(Fos*Ts)+(2*Fow*Tw)+(2*C*Ta)+(Tp*(1-Fon-Fos-(2*Fow)-(2*C)));
/* Write new value to alternative array */
if ((switchto1 == 1) && (switchto2 == 0))
{ temps2[rcnt+1][ccnt+1] = Tptp1; }
if ((switchto1 == 0) && (switchto2 == 1))
{ temps1[rcnt+1][ccnt+1] = Tptp1; }

if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
{ fprintf(table, "%9.3lf\n", Tptp1); }

return(Tptp1);
}
double leftsymm(long rcnt,long ccnt, int switchto1,int switchto2)
{
/* scan appropriate array */
if ((switchto1 == 1) && (switchto2 == 0))
{
 Tn = temps1[rcnt][ccnt+1];
 Tp = temps1[rcnt+1][ccnt+1];
 Te = temps1[rcnt+1][ccnt+2];
}
if ((switchto1 == 0) && (switchto2 == 1))
{
 Tn = temps2[rcnt][ccnt+1];
 Tp = temps2[rcnt+1][ccnt+1];
 Te = temps2[rcnt+1][ccnt+2];
}
```

```
 Kn = 15.0 + (0.013*Tn);
 Ke = 15.0 + (0.013*Te);
 density = 7900.0 - (0.5*Tp);

 Fon = (Kn*tinc)/(Elen*Elen*density*Cp);
 Foe = (Ke*tinc)/(Elen*Elen*density*Cp);

 Tptp1 = (2*Fon*Tn)+(2*Foe*Te)+(Tp*(1-(2*Fon)-(2*Foe)));
 /* Write new value to alternative array */
 if ((switchto1 == 1) && (switchto2 == 0))
 { temps2[rcnt+1][ccnt+1] = Tptp1; }
 if ((switchto1 == 0) && (switchto2 == 1))
 { temps1[rcnt+1][ccnt+1] = Tptp1; }

 if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
 {
  fprintf(table, "%4ld %6.3lf  %7.3lf", rcnt+1,(height-(rcnt*(height/Nhelements)))*1000, Tptp1);
 }

 return(Tptp1);
}
double internalsymm(long rcnt,long ccnt, int switchto1,int switchto2)
{
 /* scan appropriate array */
 if ((switchto1 == 1) && (switchto2 == 0))
 {
  Tn = temps1[rcnt][ccnt+1];
  Tw = temps1[rcnt+1][ccnt];
  Tp = temps1[rcnt+1][ccnt+1];
  Te = temps1[rcnt+1][ccnt+2];
 }
 if ((switchto1 == 0) && (switchto2 == 1))
 {
  Tn = temps2[rcnt][ccnt+1];
  Tw = temps2[rcnt+1][ccnt];
  Tp = temps2[rcnt+1][ccnt+1];
  Te = temps2[rcnt+1][ccnt+2];
 }

 Kn = 15.0 + (0.013*Tn);
 Kw = 15.0 + (0.013*Tw);
 Ke = 15.0 + (0.013*Te);
 density = 7900.0 - (0.5*Tp);

 Fon = (Kn*tinc)/(Elen*Elen*density*Cp);
 Fow = (Kw*tinc)/(Elen*Elen*density*Cp);
 Foe = (Ke*tinc)/(Elen*Elen*density*Cp);

 Tptp1 = (2*Fon*Tn)+(Foe*Te)+(Fow*Tw)+(Tp*(1-(2*Fon)-Foe-Fow));
 /* Write new value to alternative array */
 if ((switchto1 == 1) && (switchto2 == 0))
 { temps2[rcnt+1][ccnt+1] = Tptp1; }
 if ((switchto1 == 0) && (switchto2 == 1))
 { temps1[rcnt+1][ccnt+1] = Tptp1; }

 if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
 { fprintf(table, "%9.3lf", Tptp1); }

 return(Tptp1);
}
double shockedsymm(long rcnt,long ccnt, int switchto1,int switchto2)
{
 /* scan appropriate array */
 if ((switchto1 == 1) && (switchto2 == 0))
 {
  Tn = temps1[rcnt][ccnt+1];
  Tw = temps1[rcnt+1][ccnt];
```

```
 Tp = temps1[rcnt+1][ccnt+1];
 }
 if ((switchto1 == 0) && (switchto2 == 1))
 {
 Tn = temps2[rcnt][ccnt+1];
 Tw = temps2[rcnt+1][ccnt];
 Tp = temps2[rcnt+1][ccnt+1];
 }

 Kn = 15.0 + (0.013*Tn);
 Kw = 15.0 + (0.013*Tw);
 density = 7900.0 - (0.5*Tp);

 Fon = (Kn*tinc)/(Elen*Elen*density*Cp);
 Fow = (Kw*tinc)/(Elen*Elen*density*Cp);
 C = (h*tinc)/(Elen*density*Cp);

 Tptp1 = (2*Fon*Tn)+(2*Fow*Tw)+(2*C*Ta)+(Tp*(1-(2*Fon)-(2*Fow)-(2*C)));
 /* Write new value to alternative array */
 if ((switchto1 == 1) && (switchto2 == 0))
 { temps2[rcnt+1][ccnt+1] = Tptp1; }
 if ((switchto1 == 0) && (switchto2 == 1))
 { temps1[rcnt+1][ccnt+1] = Tptp1; }

 if ( (timeout >= (outinc-(tinc/2))) && (timeout <= (outinc+(tinc/2))) )
 {
  fprintf(table, "%9.3lf\n", Tptp1);
  fprintf(table, "\n\nANALYSIS INFORMATION\n");
  fprintf(table, "\nTime period(s) = %lf    Time Increment(s) = %lf\n", tperiod,tinc);
  fprintf(table, "\nSpecific Heat Capacity(J/KgK) = %.3lf", Cp);
  fprintf(table, "\nSurface Heat Transfer Coefficient(W/m2K) = %.3lf\n", h);
  fprintf(table, "\nWidth(mm) = %.3lf    Height(mm) = %.3lf", width*1000, height*1000);
  fprintf(table, "\nNumber of Width Elements   = %.0lf", Nwelements);
  fprintf(table, "\nNumber of Height Elements  = %.0lf", Nhelements);
  fprintf(table, "\nElement length(mm)         = %.3lf", Elen*1000);
  fprintf(table, "\nNumber of Shocked Elements = %.0lf", Nshelements);
  fprintf(table, "\nTotal Number of Elements   = %.0lf\n", Nwelements*Nhelements);
  fprintf(table, "\nNumber of FD Equations per Iteration = %g", Nwnodes*Nhnodes*tperiod/tinc);
  fprintf(table, "\nNumber of Iterations = %d", iternum);

  fclose(table);
  timeout = 0;
 }

 return(Tptp1);
}

/* GENERATE SUMMARY FILES */
void summarypre(double timer, double sumcnt)
{
 char name[12], csumcnt[1];
 char *ext = ".txt", *sum = "Sum";
 long column;
 int sig = 1; /* significant digits */

 gcvt(sumcnt, sig, csumcnt);
 strcpy(name, sum);
 strcat(name, csumcnt);
 strcat(name, ext);

 if ( (table = fopen(name, "w")) == NULL )
 {
 printf("\n*** Cannot open file %s ***\n", name);
 exit(0);
 }

 fprintf(table, "**********\n");
```

```
 fprintf(table, "Time = %.30lf\n", timer);
 fprintf(table, "*************\n");
 fprintf(table, "     COLUMN");
 column = 1;
 while (column <= Nwnodes)
 {
  if (column < Nwnodes)
  { fprintf(table, "%9d", column); }
  if (column == Nwnodes)
  { fprintf(table, "%9d\n", column); }
  column ++;
 }
 column = 1;
 fprintf(table, " ROW POSTN.");
 while (column <= Nwnodes)
 {
  if (column < Nwnodes)
  { fprintf(table, "%9.3lf", (width-((column-1)*(width/Nwelements)))*1000); }
  if (column == Nwnodes)
  { fprintf(table, "%9.3lf\n", (width-((column-1)*(width/Nwelements)))*1000); }
  column ++;
 }

 return;
}

/* PRINT ITERATION SUMMARY TO SCREEN AND LOG FILE */
void itersummary()
{
 fprintf(ilog, "**********************************\n");
 fprintf(ilog, "ITERATION NO. %d\n", iternum);
 fprintf(ilog, "Time increment = %.3lf,  (max = %.3lf)\n", oldtinc, oldminc);
 fprintf(ilog, "Surface Heat Transfer Coefficient = %.3lf\n", oldh);
 fprintf(ilog, "Fourier Number = %.3lf\n", oldFo);
 fprintf(ilog, "After %.3lf seconds\n", tperiod);
 fprintf(ilog, "Shocked centre Temperature = %.3lfC\n", Tshock);
 fprintf(ilog, "Temperature error = %.3lfC\n", Terror);
 fprintf(ilog, "Temperature error factor = %lf\n", Tfac);
 if ( (Tfac <= -0.0009) || (Tfac >= 0.0009) )
 {
  fprintf(ilog, "Next Surface Heat Transfer Coefficient = %.3lf\n", h);
 }
 fprintf(ilog, "Number of shocked nodes = %.0lf\n", Nshelements+1);
 fprintf(ilog, "**********************************\n\n");

/* fprintf(fp, "End of iteration %d\n\n", iternum); */

 if ( (prntoscrn == 'Y') || (prntoscrn == 'y') )
 {
  printf("**********************************\n");
  printf("ITERATION NO. %d\n", iternum);
  printf("Time increment = %.3lf,  (max = %.3lf)\n", oldtinc, oldminc);
  printf("Surface Heat Transfer Coefficient = %.3lf\n", oldh);
  printf("Fourier Number = %.3lf\n", oldFo);
  printf("After %.3lf seconds\n", tperiod);
  printf("Shocked centre Temperature = %.3lfC\n", Tshock);
  printf("Temperature error = %.3lfC\n", Terror);
  printf("Temperature error factor = %lf\n", Tfac);
 if ( (Tfac <= -0.0009) || (Tfac >= 0.0009) )
 {
  printf("Next Surface Heat Transfer Coefficient = %.3lf\n", h);
 }
  printf("Number of shocked nodes = %.0lf\n", Nshelements+1);
  printf("**********************************\n\n");
 }
 else
 {
```

```
 printf("ITERATION NO. %d    Shocked Temperature = %.3lf\n", iternum, Tshock);
 printf("Time = %.30lf\n", timer-tinc);
 }

 return;
}

void timerstart()
{
 time_t t;

 time(&t);
 printf("Programme Began: %s\n", ctime(&t));
 fprintf(ilog, "Programme Began: %s\n\n", ctime(&t));
}
void timerend()
{
 time_t t;

 time(&t);
 printf("\n\nProgramme Completed: %s\n", ctime(&t));
 fprintf(ilog, "\n\nProgramme Completed: %s\n", ctime(&t));
}
```

Appendix B.2

Programme Code: *UDISP* Fortran Code

Exponential Reheat for Return Thermal Shock Cycle by;

**BOUNDARY, UDISP User Routine*

```
SUBROUTINE DISP(U,KSTEP,KINC,TIME,NODE,JDOF)
C
    INCLUDE 'ABA_PARAM.INC'
C
    DIMENSION U(3),TIME(2)
C
C
    TMAX = 625
C
C REHEAT EXPONENTIAL EQUATION
    U(1) = U(1) + ((TMAX-U(1))*(1-EXP(-0.125*TIME(1))))
C
C
    RETURN
    END
```

Appendix B.3

Programme Code: Thermal Shock Analysis Mesh Generator

Mesh Generator for Thermal Shock Fracture Model Configuration

(Stress Analysis)

```
/* Mesh Generator - Multiple Files
   Thermal Shock Variable Crack Length */

#include <stdio.h>
#include <math.h>

main()
{
 char  reply;
 int   Elsh,Elsw,con, Elaft,Elaftmin,Elbef,Elabov,Elabovmin;
 int   shintervals;
 int   crcnt,ncracks;
 float shreg;
 float crinc;
 float SElh,SElw, height,width,theight;
 float x, postn, priormd,aftermd,abovemd, singh,singw;
 float biasu, afirstElen,alastElen,bfirstElen,blastElen,ufirstElen,ulastElen,
       sum;
 float biaso, aratio,bratio,uratio,fratio, distance, aerror,berror,uerror;
 double Elbias, cnt;

 FILE *fp;


 printf("\n\n**********************************\n");
 printf("**Programme: shockmesh.c\n");
 printf("**MESH DATA GENERATION programme\n");
 printf("  To calculate required model components");
 printf("  CREATE `datasum' FILE");
 printf("\n**********************************\n");


 height = 106.0;
 width = 40.0;

/* Make Geometry and Element Requests */
 printf("\n** Geometry Information **\n");
 printf("Crack Tip Position = ");
 scanf(" %f", &postn);
 printf("Singularity Mesh Width (mm)\n");
 printf("Maximum Width Possible With Current Crack Length is %.3f: ", postn*width*2);
 scanf(" %f", &singw);
 printf("Singularity Mesh Height (mm)= ");
 scanf(" %f", &singh);
 printf("Height of Shocked Region for Upper Crack Edge (mm)= ");
 scanf(" %f", &shreg);
 if (shreg < singh)
 {
  printf("***WARNING - The shocked region is smaller than the singularity height\n");
  printf("             You will have to define the region yourself\n");
 }

/* Test Crack and Singularity Geometry */
 while ((singw/2) > (width*postn))
 {
  printf("*** Error: Half the specified width, %.3fmm, is greater than the crack length, %.3fmm ***\n",
singw/2,width*postn);
  printf("Crack Tip Position = ");
  scanf(" %f", &postn);
  printf("Singularity Mesh Height (mm)= ");
  scanf(" %f", &singh);
  printf("Singularity Mesh Width (mm)= ");
  scanf(" %f", &singw);
 }

/* Open Crack Length minus Singularity Mesh */
 priormd =  (postn*width)-(singw/2);
```

```
 if (singw == postn*width*2) { priormd = 0; }
/* Ligament Length minus Singularity Mesh*/
 aftermd = ((1-postn)*width)-(singw/2);
/* Distance Above Singularity to tied Surface */
 abovemd = height-singh;

 printf("\n** Element Information **\n");
 printf("No. of Contours = ");
 scanf(" %d", &con);
 printf("No. of Elements Up Singularity Mesh = ");
 scanf(" %d", &Elsh);
 printf("No. of Elements Across Singularity Mesh = ");
 scanf(" %d", &Elsw);
 printf("\nMinimum No. of Elements After Crack = ");
 scanf(" %d", &Elaftmin);

 printf("\n** Biasing - only takes place outside singularity **\n");
 printf("Suggested Outward Bias = ");
 scanf(" %f", &biaso);
 while (biaso >= 1.0)
 {
  printf("Bias must be less than 1.0\n");
  printf("Suggested Outward Bias = ");
  scanf(" %f", &biaso);
 }

 printf("\nMinimum No. of Elements Upwards:\n");
 printf("Suggested Region - %.1f: Required Minimum = ", (abovemd/aftermd)*Elaftmin);
 scanf(" %d", &Elabovmin);
 printf("Suggested Upward Bias = ");
 scanf(" %f", &biasu);
 while (biasu >= 1.0)
 {
  printf("Bias must be less than 1.0\n");
  printf("Suggested Upward Bias = ");
  scanf(" %f", &biasu);
 }

/* printf("Upward Bias taken as %.3f\n\n\n", biaso);
 biasu = biaso;*/

/* Singularity Bulk Element Size */
 SElh = singh/Elsh;
 SElw = singw/Elsw;

/* After Crack Outward Biasing and No. of Elements Calculation */
 aerror = aftermd+1;
 aratio = 8.1;
 while (aratio > 8.0)
 {
  aerror = 0.06*aftermd;
  while ((aerror > (0.05*aftermd)))
  {
   cnt = 0;
   sum = 0;
   distance = 0;
   Elaft = Elaftmin-1;
   while ((distance <= aftermd) || (Elaft < Elaftmin))
   {
    Elbias = pow(biaso,cnt);
    sum = sum + (1/Elbias);
    afirstElen = SElh;
    alastElen = afirstElen/Elbias;
    distance = afirstElen*sum;
    Elaft = cnt+1;
    cnt++;
   }
```

```
  aerror = distance - aftermd;
  biaso = biaso + 0.005;
  }
  aratio = alastElen/afirstElen;
  biaso = biaso - 0.001;
 }
 biaso = biaso - 0.004;

/* Open data summary file for writing required information */
 fp = fopen("datasum", "w");

/* Output Data For After Tip Mesh Generation */

 printf("\n\n***Final Output data for after tip mesh generation***\n");
 printf("First Element Length is %.3f Last Element Length is %.3f\n", afirstElen,alastElen);
 printf("Distance Error = %.3f\n", aerror);
 printf("First to Last Element Ratio = %.3f\n", aratio);
 printf("Outward Biasing = %.3f\n", biaso);
 printf("No. of Elements in after crack mesh = %d\n\n", Elaft);


/* Pre-Crack Mesh Generation */
 Elbef = 0;
 if (priormd != 0)
 {
  cnt = 0;
  sum = 0;
  distance = 0;
  Elbef = 0;
  while (distance < priormd)
  {
   Elbias = pow(biaso,cnt);
   sum = sum + (1/Elbias);
   bfirstElen = SElh;
   blastElen = bfirstElen/Elbias;
   distance = bfirstElen*sum;
   Elbef = cnt+1;
   cnt++;
  }
  berror = distance - priormd;
  bratio = blastElen/bfirstElen;

 /* Output Data For Before Tip Mesh Generation */
   printf("***Final Output data for before tip mesh generation***\n");
   printf("Prior mesh distance = %f\n", priormd);
   printf("First Element Length is %.3f Last Element Length is %.3f\n", bfirstElen,blastElen);
   printf("Distance Error = %.3f\n", berror);
   printf("First to Last Element Ratio = %.3f\n", bratio);
   printf("No. of Elements in prior crack mesh = %d\n\n", Elbef);
 }

/* Upward Biasing and No. of Elements Calculation */
 uerror = abovemd+1;
 uratio = 25.1;
 while (uratio > 25.0)
 {
  uerror = 0.06*abovemd;
  while ((uerror > (0.05*abovemd)))
  {
   cnt = 0;
   sum = 0;
   distance = 0;
   Elabov = Elabovmin-1;
   while ((distance <= abovemd) || (Elabov < Elabovmin))
   {
    Elbias = pow(biasu,cnt);
    sum = sum + (1/Elbias);
```

```
  ufirstElen = SElw;
  ulastElen = ufirstElen/Elbias;
  distance = ufirstElen*sum;
  if (distance <= (shreg-singh))
  {
   shintervals = cnt+1;
  }
  Elabov = cnt+1;
  cnt++;
 }
 uerror = distance - abovemd;
 biasu = biasu + 0.005;
}
uratio = ulastElen/ufirstElen;
biasu = biasu - 0.001;
}
biasu = biasu - 0.004;

/* Output Data For Above Tip Mesh Generation */

printf("***Final Output data for above tip mesh generation***\n");
printf("First Element Length is %.3f Last Element Length is %.3f\n", ufirstElen,ulastElen);
printf("Distance Error = %.3f\n", uerror);
printf("First to Last Element Ratio = %.3f\n", uratio);
printf("Upward Biasing = %.3f\n", biasu);
printf("No. of Elements above singularity mesh to top surface = %d\n", Elabov);
printf("No. of Elements above singularity mesh to shock limit = %d\n\n", shintervals);


/* Write Element Givens to data summary file */
fprintf(fp, "*** Fixed Model Data ***\n");
fprintf(fp, "Crack Tip Position (a/w)    = %.3f\n", postn);
fprintf(fp, "Singularity Height is       = %.3f\n", singh);
fprintf(fp, "Singularity Width is        = %.3f\n", singw);
fprintf(fp, "Number of J-Int. Contours   = %d\n", con);
fprintf(fp, "Elements Up Singularity     = %d\n", Elsh);
fprintf(fp, "Elements Across Singularity = %d\n", Elsw);
fprintf(fp, "Calculated Outward Biasing  = %.3f\n", biaso);
fprintf(fp, "Calculated Upward Biasing   = %.3f\n", biasu);
fprintf(fp, "Number of Shocked Elements  = %d\n", shintervals);


/* Begin Repeated Crack Length Model Generation */

printf("Would You Like to Generate Further Crack Lengths?\n");
printf("Cracks will use quoted biasing and advance on original length (y/n): ");
scanf(" %c", &reply);

if ((reply == 'y') || (reply == 'Y'))
{
 printf("\nWhat is the a/w increment? ");
 scanf(" %f", &crinc);
 printf("How Many Crack Lengths are Required? ");
 scanf(" %d", &ncracks);

 /* Write Number Of Crack Lengths to data summary file */
 fprintf(fp, "Number of Crack Lengths     = %d\n", ncracks);
 fprintf(fp, "Crack Length Increment      = %.3f\n\n", crinc);
}

if ((reply !='y') && (reply != 'Y'))
{ fprintf(fp, "Number of Crack Lengths     = 1\n\n"); }

fprintf(fp, "***\n");
fprintf(fp, "*** Crack 1 Data ***  a/w = %.3f\n", postn);
fprintf(fp, "Number of Afterward Elements = %d\n", Elaft);
fprintf(fp, "Number of Prior Elements     = %d\n", Elbef);
```

```
fprintf(fp, "Number of Upward Elements    = %d\n", Elabov);

if ((reply == 'y') || (reply == 'Y'))
{
 crcnt = 1;
 while (crcnt < ncracks)
 {
 /*After Tip Element Calculation */
 cnt = 0;
 sum = 0;
 distance = 0;
 Elaft = Elaftmin-1;
 while (distance <= aftermd)
 {
 aftermd = ((1-(postn+(crcnt*crinc)))*width)-(singw/2);
 Elbias = pow(biaso,cnt);
 sum = sum + (1/Elbias);
 afirstElen = SElh;
 alastElen = afirstElen/Elbias;
 distance = afirstElen*sum;
 Elaft = cnt+1;
 cnt++;
 }

 /* Before Tip Element Calculation */
 if (priormd > 0.0)
 {
 cnt = 0;
 sum = 0;
 distance = 0;
 Elbef = 0;
 while (distance <= priormd)
 {
  priormd =  ((postn+(crcnt*crinc))*width)-(singw/2);
  Elbias = pow(biaso,cnt);
  sum = sum + (1/Elbias);
  bfirstElen = SElh;
  blastElen = bfirstElen/Elbias;
  distance = bfirstElen*sum;
  Elbef = cnt+1;
  cnt++;
 }
 }

 /* Abaove Tip Element Calculation */
 cnt = 0;
 sum = 0;
 distance = 0;
 Elabov = Elabovmin-1;
 while ((distance <= abovemd) || (Elabov < Elabovmin))
 {
 Elbias = pow(biasu,cnt);
 sum = sum + (1/Elbias);
 ufirstElen = SElw;
 ulastElen = ufirstElen/Elbias;
 distance = ufirstElen*sum;
 Elabov = cnt+1;
 cnt++;
 }
 fprintf(fp, "***\n");
 fprintf(fp, "*** Crack %d Data ***  a/w = %.3f\n", crcnt+1, crinc*(crcnt+1));
 fprintf(fp, "Number of Afterward Elements = %d\n", Elaft);
 fprintf(fp, "Number of Prior Elements     = %d\n", Elbef);
 fprintf(fp, "Number of Upward Elements    = %d\n", Elabov);
 crcnt++;
 }
}
```

Appendix B.4

Programme Code: Cruciform Analysis Mesh Generator

Mesh Generator for Cruciform Fracture Model Configuration

```
/* Mesh Generator - Multiple Files
  Cruciform Variable Crack Length */

#include <stdio.h>
#include <math.h>

main()
{
 char  reply;
 int   Elsh,Elsw,con, Elaft,Elaftmin,Elbef,Elabov,Elabovmin;
 int   crcnt,ncracks;
 float  crinc;
 float  SElh,SElw, height,width;
 float  x, postn, priormd,aftermd,abovemd, singh,singw;
 float  biasu, afirstElen,alastElen,bfirstElen,blastElen,ufirstElen,ulastElen,
      sum;
 float  biaso, aratio,bratio,uratio,fratio, distance, aerror,berror,uerror;
 double Elbias, cnt;

 FILE *fp;

 printf("\n\n*****************************************************\n");
 printf("Multiple Crack Length Input File Generator\n");
 printf("C Programme: meshgen.c\n");
 printf("Creates a file called 'datasum' in current directory\n");
 printf("C Programme: progen.c requires this file to run\n");
 printf("*****************************************************\n");


 height = 50.0;
 width = 50.0;

/* Make Geometry and Element Requests */
 printf("\n** Geometry Information **\n");
 printf("Crack Tip Position = ");
 scanf(" %f", &postn);
 printf("Singularity Mesh Width (mm)\n");
 printf("Maximum Width Possible With Current Crack Length is %.3f: ", postn*width*2);
 scanf(" %f", &singw);
 printf("Singularity Mesh Height (mm)= ");
 scanf(" %f", &singh);

/* Test Crack and Singularity Geometry */
 while ((singw/2) > (width*postn))
 {
  printf("*** Error: Half the specified width, %.3fmm, is greater than the crack length, %.3fmm ***\n",
singw/2,width*postn);
  printf("Crack Tip Position = ");
  scanf(" %f", &postn);
  printf("Singularity Mesh Height (mm)= ");
  scanf(" %f", &singh);
  printf("Singularity Mesh Width (mm)= ");
  scanf(" %f", &singw);
 }

/* Open Crack Length minus Singularity Mesh */
 priormd =  (postn*width)-(singw/2);
/* Ligament Length minus Singularity Mesh*/
 aftermd = ((1-postn)*width)-(singw/2);
/* Distance Above Singularity to tied Surface */
 abovemd = height-singh;

 printf("\n** Element Information **\n");
 printf("No. of Contours = ");
 scanf(" %d", &con);
 printf("No. of Elements Up Singularity Mesh = ");
 scanf(" %d", &Elsh);
```

```
printf("No. of Elements Across Singularity Mesh = ");
scanf(" %d", &Elsw);
printf("\nMinimum No. of Elements After Crack = ");
scanf(" %d", &Elaftmin);

printf("\n** Biasing - only takes place outside singularity **\n");
printf("Suggested Outward Bias = ");
scanf(" %f", &biaso);
while (biaso >= 1.0)
{
 printf("Bias must be less than 1.0\n");
 printf("Suggested Outward Bias = ");
 scanf(" %f", &biaso);
}

printf("\nMinimum No. of Elements Upwards to Tied Surface:\n");
printf("Suggested Region - %.1f: Required Minimum = ", (abovemd/aftermd)*Elaftmin);
scanf(" %d", &Elabovmin);
printf("Suggested Upward Bias = ");
scanf(" %f", &biasu);
while (biasu >= 1.0)
{
 printf("Bias must be less than 1.0\n");
 printf("Suggested Upward Bias = ");
 scanf(" %f", &biasu);
}

/* printf("Upward Bias taken as %.3f\n\n\n", biaso);
 biasu = biaso;*/

/* Singularity Bulk Element Size */
 SElh = singh/Elsh;
 SElw = singw/Elsw;

/* After Crack Outward Biasing and No. of Elements Calculation */
 aerror = aftermd+1;
 aratio = 4.1;
 while (aratio > 4.0)
 {
  aerror = 0.06*aftermd;
  while ((aerror > (0.05*aftermd)))
  {
   cnt = 0;
   sum = 0;
   distance = 0;
   Elaft = Elaftmin-1;
   while ((distance <= aftermd) || (Elaft < Elaftmin))
   {
    Elbias = pow(biaso,cnt);
    sum = sum + (1/Elbias);
    afirstElen = SElh;
    alastElen = afirstElen/Elbias;
    distance = afirstElen*sum;
    Elaft = cnt+1;
    cnt++;
   }
   aerror = distance - aftermd;
   biaso = biaso + 0.005;
  }
  aratio = alastElen/afirstElen;
  biaso = biaso - 0.001;
 }
 biaso = biaso - 0.004;

/* Open data summary file for writing required information */
 fp = fopen("datasum", "w");
```

```
/* Output Data For After Tip Mesh Generation */

 printf("\n\n***Final Output data for after tip mesh generation***\n");
 printf("First Element Length is %.3f Last Element Length is %.3f\n", afirstElen,alastElen);
 printf("Distance Error = %.3f\n", aerror);
 printf("First to Last Element Ratio = %.3f\n", aratio);
 printf("Outward Biasing = %.3f\n", biaso);
 printf("No. of Elements in after crack mesh = %d\n\n", Elaft);


/* Pre-Crack Mesh Generation */
 Elbef = 0;
 if (priormd > 0.0)
 {
  cnt = 0;
  sum = 0;
  distance = 0;
  Elbef = 0;
  while (distance <= priormd)
  {
   Elbias = pow(biaso,cnt);
   sum = sum + (1/Elbias);
   bfirstElen = SElh;
   blastElen = bfirstElen/Elbias;
   distance = bfirstElen*sum;
   Elbef = cnt+1;
   cnt++;
  }
  berror = distance - priormd;
  bratio = blastElen/bfirstElen;

 /* Output Data For Before Tip Mesh Generation */

  printf("***Final Output data for before tip mesh generation***\n");
  printf("First Element Length is %.3f Last Element Length is %.3f\n", bfirstElen,blastElen);
  printf("Distance Error = %.3f\n", berror);
  printf("First to Last Element Ratio = %.3f\n", bratio);
  printf("No. of Elements in prior crack mesh = %d\n\n", Elbef);
 }

/* Upward Biasing and No. of Elements Calculation */
 uerror = abovemd+1;
 uratio = 4.1;
 while (uratio > 4.0)
 {
  uerror = 0.06*abovemd;
  while ((uerror > (0.05*abovemd)))
  {
   cnt = 0;
   sum = 0;
   distance = 0;
   Elabov = Elabovmin-1;
   while ((distance <= abovemd) || (Elabov < Elabovmin))
   {
    Elbias = pow(biasu,cnt);
    sum = sum + (1/Elbias);
    ufirstElen = SElw;
    ulastElen = ufirstElen/Elbias;
    distance = ufirstElen*sum;
    Elabov = cnt+1;
    cnt++;
   }
   uerror = distance - abovemd;
   biasu = biasu + 0.005;
  }
  uratio = ulastElen/ufirstElen;
  biasu = biasu - 0.001;
```

```
 }
 biasu = biasu - 0.004;

/* Output Data For Above Tip Mesh Generation */

 printf("***Final Output data for above tip mesh generation***\n");
 printf("First Element Length is %.3f Last Element Length is %.3f\n", ufirstElen,ulastElen);
 printf("Distance Error = %.3f\n", uerror);
 printf("First to Last Element Ratio = %.3f\n", uratio);
 printf("Upward Biasing = %.3f\n", biasu);
 printf("No. of Elements above singularity mesh to tied surface = %d\n\n", Elabov);


/* Write Element Givens to data summary file */
 fprintf(fp, "*** Fixed Model Data ***\n");
 fprintf(fp, "Crack Tip Position (a/w)    = %.3f\n", postn);
 fprintf(fp, "Singularity Height is       = %.3f\n", singh);
 fprintf(fp, "Singularity Width is        = %.3f\n", singw);
 fprintf(fp, "Number of J-Int. Contours   = %d\n", con);
 fprintf(fp, "Elements Up Singularity     = %d\n", Elsh);
 fprintf(fp, "Elements Across Singularity = %d\n", Elsw);
 fprintf(fp, "Calculated Outward Biasing  = %.3f\n", biaso);
 fprintf(fp, "Calculated Upward Biasing   = %.3f\n", biasu);


/* Begin Repeated Crack Length Model Generation */

 printf("Would You Like to Generate Further Crack Lengths?\n");
 printf("Cracks will use quoted biasing and advance on original length (y/n): ");
 scanf(" %c", &reply);

 if ((reply == 'y') || (reply == 'Y'))
 {
  printf("\nWhat is the a/w increment? ");
  scanf(" %f", &crinc);
  printf("How Many Crack Lengths are Required? ");
  scanf(" %d", &ncracks);

  /* Write Number Of Crack Lengths to data summary file */
  fprintf(fp, "Number of Crack Lengths     = %d\n", ncracks);
  fprintf(fp, "Crack Length Increment      = %.3f\n\n", crinc);
 }

 if ((reply !='y') && (reply != 'Y'))
 { fprintf(fp, "Number of Crack Lengths     = 1\n\n"); }

 fprintf(fp, "***\n");
 fprintf(fp, "*** Crack 1 Data ***\n");
 fprintf(fp, "Number of Afterward Elements = %d\n", Elaft);
 fprintf(fp, "Number of Prior Elements     = %d\n", Elbef);
 fprintf(fp, "Number of Upward Elements    = %d\n", Elabov);

 if ((reply == 'y') || (reply == 'Y'))
 {
  crcnt = 1;
  while (crcnt < ncracks)
  {
   /*After Tip Element Calculation */
   cnt = 0;
   sum = 0;
   distance = 0;
   Elaft = Elaftmin-1;
   while (distance <= aftermd)
   {
    aftermd = ((1-(postn+(crcnt*crinc)))*width)-(singw/2);
    Elbias = pow(biaso,cnt);
    sum = sum + (1/Elbias);
```

```
   afirstElen = SElh;
   alastElen = afirstElen/Elbias;
   distance = afirstElen*sum;
   Elaft = cnt+1;
   cnt++;
   }

  /* Before Tip Element Calculation */
  if (priormd > 0.0)
  {
   cnt = 0;
   sum = 0;
   distance = 0;
   Elbef = 0;
   while (distance <= priormd)
   {
    priormd =  ((postn+(crcnt*crinc))*width)-(singw/2);
    Elbias = pow(biaso,cnt);
    sum = sum + (1/Elbias);
    bfirstElen = SElh;
    blastElen = bfirstElen/Elbias;
    distance = bfirstElen*sum;
    Elbef = cnt+1;
    cnt++;
   }
  }

  /* Abaove Tip Element Calculation */
  cnt = 0;
  sum = 0;
  distance = 0;
  Elabov = Elabovmin-1;
  while ((distance <= abovemd) || (Elabov < Elabovmin))
  {
   Elbias = pow(biasu,cnt);
   sum = sum + (1/Elbias);
   ufirstElen = SElw;
   ulastElen = ufirstElen/Elbias;
   distance = ufirstElen*sum;
   Elabov = cnt+1;
            cnt++;
  }
  fprintf(fp, "***\n");
  fprintf(fp, "*** Crack %d Data ***\n", crcnt+1);
  fprintf(fp, "Number of Afterward Elements = %d\n", Elaft);
  fprintf(fp, "Number of Prior Elements     = %d\n", Elbef);
  fprintf(fp, "Number of Upward Elements    = %d\n", Elabov);
  crcnt++;
 }
}


printf("\n\nI'm Done\n\n");

fclose(fp);
return;
}
```

Appendix B.5

Programme Code: Thermal Shock Input File Generator

ABAQUS Input File Generator for Thermal Shock Fracture Model

(Stress Analysis)

```
/* Input File Generator - Multiple Files
  Thermal Shock Variable Crack Length */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

main()
{
 char  postnc[5], fname[20],fnamex[20], fadd[10];
 char  Elaftc[3],Elbefc[3],Elabovc[3], Elshc[3],Elswc[3], conc[2];
 char  biasoc[5],biasuc[5], singhc[4],singwc[4];
 char  ncracksc[2],crincc[5];
 int   inChar;
 int   Elaft,Elbef,Elabov, Elsh,Elsw, con;
 int   Sn1,Sn2,Sn3,Sn4,Sn5,Sn6,Sn7,Sn8,Sn9, Sinc;
 int   Bn1,Bn2,Bn3,Bn4,Bn5,Bn6,Bn7,Bn8, Bn9,Bn10,Bn11,Bn12, Binc, Tinc;
 int   S, Ba,Bb,Bc,Bd,Be, temp,tcheck, line;
 int   ncracks, crcnt;
 long  cnt;
 float postn, singh,singw, priormd;
 float biaso,biasu, Srange,Smean, fx2,fx3,fy5,fx7,fx8;
 float sx1,sy1, sx2,sy2, sx3,sy3, sx4,sy4, sx5,sy5,
     sx6,sy6, sx7,sy7, sx8,sy8, sx9,sy9,
     bx1,by1, bx2,by2, bx3,by3, bx4,by4, bx5,by5,
     bx6,by6, bx7,by7, bx8,by8, bx9,by9;
 float crinc;
 float height, width;

 FILE *read;
 FILE *fp;
 FILE *mat;


 height = 30.0;
 width = 40.0;


/* Open data summary file for reading */
 if ((read = fopen("datasum", "r")) == NULL)
 { printf("Cannot open datasum\n"); }

/* Read & Write Fixed Element Data */
 cnt = 31;

 fseek(read, 25+(1*cnt), 0);
 fscanf(read, "%s", postnc);
 postn = atof(postnc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", singhc);
 singh = atof(singhc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", singwc);
 singw = atof(singwc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", conc);
 con = atoi(conc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elshc);
 Elsh = atoi(Elshc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elswc);
```

```
Elsw = atof(Elswc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", biasoc);
biaso = atof(biasoc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", biasuc);
biasu = atof(biasuc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", ncracksc);
ncracks = atoi(ncracksc);

if (ncracks > 1)
{
 fseek(read, cnt+1, 1);
 fscanf(read, "%s", crincc);
 crinc = atof(crincc);
 }

if (ncracks == 1)
{
 crinc = 0;
 fseek(read, 27+cnt, 1);
 fscanf(read, "%s", Elaftc);
 Elaft = atoi(Elaftc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elbefc);
 Elbef = atoi(Elbefc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elabovc);
 Elabov = atoi(Elabovc);
 }

if (ncracks > 1)
{
 fseek(read, 26+1+cnt, 1);
 fscanf(read, "%s", Elaftc);
 Elaft = atoi(Elaftc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elbefc);
 Elbef = atoi(Elbefc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elabovc);
 Elabov = atoi(Elabovc);
 }

/* Define Singulrity Nodes */
 Sinc = (2*con)+(2*Elaft)+1;
 Sn1 = 1;
 Sn2 = Sn1+(2*con);
 Sn3 = Sn2+(Elsh*Sinc);
 Sn4 = Sn3+(Elsh*Sinc);
 Sn5 = Sn4+(Elsw*Sinc);
 Sn6 = Sn5+(Elsw*Sinc);
 Sn7 = Sn6+(Elsh*Sinc);
 Sn8 = Sn7+(Elsh*Sinc);
 Sn9 = Sn8-(2*con);

 printf("What is the Cyclic Stress Range (MPa, R=0.1): ");
 scanf(" %f", &Srange);
 Smean = (0.45*(Srange/0.9))+(0.1*(Srange/0.9));
```

```
printf("What is the Temperature (celcius): ");
scanf(" %d", &temp);
printf("\n");

tcheck = 0;
while( tcheck == 0)
{
 if (temp == 300)  { tcheck = 1; }
 if (temp == 350)  { tcheck = 1; }
 if (temp == 400)  { tcheck = 1; }
 if (temp == 450)  { tcheck = 1; }
 if (temp == 500)  { tcheck = 1; }
 if (temp == 550)  { tcheck = 1; }
 if (temp == 600)  { tcheck = 1; }
 if (temp == 625)  { tcheck = 1; }
 if (temp == 650)  { tcheck = 1; }
 if (temp == 700)  { tcheck = 1; }

 if (tcheck == 0)
 {
  printf("*** ERROR: Temperatures are only available for:\n");
  printf("        300, 350, 400, 450, 500, 550, 600, 625, 650, 700\n\n");
  printf("Please provide one the available Temperatures: ");
       scanf(" %d", &temp);
       printf("\n");
 }
}


/* Start of Programme Loop For Multiple Crack Length Files */
crcnt = 0;
while (crcnt < ncracks)
{
 if ( ncracks > 1)
 {
 printf("Provide a file name for crack %d at a/w = %.3f: ", crcnt+1,postn);
 scanf(" %s", fname);
 strcat(fname, ".inp");
 if ((fp = fopen(fname, "w")) == NULL)
 { printf("Cannot open %s\n", fname); }
 }
 if ( ncracks == 1)
 {
 if ((fp = fopen("input.inp", "w")) == NULL)
 { printf("Cannot open file input.inp\n"); }
 }

 /* Define Body Nodes */
 Bn1 = Sn2+(2*Elbef);
 Bn2 = Bn1+(2*Elsh*Sinc);
 Bn8 = Sn8+(2*Elaft);
 Binc = Bn8-Sn4+1;

 printf("Minimum Nodal Increment Upwards is %d\n", Binc);
 printf("Provide an increment greater or equal to this: ");
 scanf(" %d", &Tinc);
 while (Tinc < Binc)
 {
  printf("Increment must be greater than %d\n", Binc);
  printf("New Upwards Increment = ");
  scanf(" %d", &Tinc);
  printf("\n");
 }
 Binc = Tinc;

 Bn3 = Bn2+(2*Elabov*Binc);
 Bn4 = Bn3-(2*Elbef);
```

```
Bn5 = Bn4+(2*Elsw*Sinc);
Bn6 = Bn5+(2*Elaft);
Bn7 = Sn6+(2*Elaft);
Bn8 = Sn8+(2*Elaft);

priormd = (postn*width)-(singw/2);
printf("Prior Mesh Distance is %.3f\n\n", priormd);

fx2 = singh*0.25;
if (fx2 >= 0.40*priormd) { fx2 = 0.40*priormd; }
fx3 = singh*0.175;
if (fx3 >= 0.25*priormd) { fx3 = 0.25*priormd; }
fy5 = singw*0.15;
fx7 = singh*0.175;
fx8 = singh*0.25;


/* Singularity node co-ordinates */
sx1 = width*postn;
sy1 = 0.0;
sx2 = (width*postn)-(singw/2)-fx2;
if (Elbef == 0) { sx2 = 0.0; }
sy2 = 0.0;
sx3 = (width*postn)-(singw/2)-fx3;
if (Elbef == 0) { sx3 = 0.0; }
sy3 = singh/2;
sx4 = (width*postn)-(singw/2);
if (Elbef == 0) { sx4 = 0.0; }
sy4 = singh;
sx5 = (width*postn);
sy5 = singh+fy5;
sx6 = (width*postn)+(singw/2);
sy6 = singh;
sx7 = (width*postn)+(singw/2)+fx7;
sy7 = singh/2;
sx8 = (width*postn)+(singw/2)+fx8;
sy8 = 0.0;
sx9 = width*postn;
sy9 = 0.0;

/* Body node co-ordinates */
bx1 = 0.0;
by1 = 0.0;
bx2 = 0.0;
by2 = singh;
bx3 = 0.0;
by3 = height;
bx4 = (width*postn)-(singw/2);
by4 = height;
bx5 = (width*postn)+(singw/2);
by5 = height;
bx6 = width;
by6 = height;
bx7 = width;
by7 = singh;
bx8 = width;
by8 = 0.0;


/* Generate Abaqus Input File */

fprintf(fp, "*HEADING\n");
fprintf(fp, " ISOTHERMAL - %dC\n", temp);
fprintf(fp, " EQUIBIAXIAL - CYCLIC STRESS RANGE = %.1fMPa\n", Srange);
fprintf(fp, " %.3f A/W CRACK LENGTH\n", postn);
fprintf(fp, "**\n");
fprintf(fp, "*RESTART, WRITE, FREQUENCY=5\n");
```

```
 fprintf(fp, "*PREPRINT, ECHO=NO, MODEL=NO\n");
 fprintf(fp, "**\n");
 fprintf(fp, "**NODE GENERATION\n");
 fprintf(fp, "**SINGULARITY\n");
 fprintf(fp, "*NODE\n");
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn1, sx1,sy1);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn2, sx2,sy2);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn3, sx3,sy3);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn4, sx4,sy4);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn5, sx5,sy5);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn6, sx6,sy6);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn7, sx7,sy7);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn8, sx8,sy8);
 fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn9, sx9,sy9);
 fprintf(fp, "*NGEN, LINE=P, NSET=SINGRITE\n");
 fprintf(fp, " %d,%d, %d,%d\n", Sn2,Sn4, Sinc,Sn3);
 fprintf(fp, "*NGEN, LINE=P, NSET=SINGTOP\n");
 fprintf(fp, " %d,%d, %d,%d\n", Sn4,Sn6, Sinc,Sn5);
 fprintf(fp, "*NGEN, LINE=P, NSET=SINGLEFT\n");
 fprintf(fp, " %d,%d, %d,%d\n", Sn6,Sn8, Sinc,Sn7);
 fprintf(fp, "*NGEN, NSET=TIP\n");
 fprintf(fp, " %d,%d, %d\n", Sn1,Sn9, Sinc);
 fprintf(fp, "*NSET, NSET=SINGSURR\n");
 fprintf(fp, " SINGRITE,SINGTOP,SINGLEFT\n");
 fprintf(fp, "*NFILL, NSET=NSING,SINGULAR=1\n");
 fprintf(fp, " TIP,SINGSURR, %d\n", 2*con,1);
 fprintf(fp, "**\n");
 fprintf(fp, "**BODY SURROUNDINGS\n");
 fprintf(fp, "*NODE\n");
 if (Elbef != 0)
 {
  fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn1,bx1,by1);
  fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn2,bx2,by2);
  fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn3,bx3,by3);
 }
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn4,-bx4,by4);
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn5,-bx5,by5);
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn6,-bx6,by6);
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn7,-bx7,by7);
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn8,-bx8,by8);
 fprintf(fp, "*NGEN, NSET=RITELOW\n");
 fprintf(fp, " %d,%d, %d\n", Bn1,Bn2, Sinc);
 fprintf(fp, "*NGEN, NSET=LEFTLOW\n");
 fprintf(fp, " %d,%d, %d\n", Bn7,Bn8, Sinc);
 fprintf(fp, "*NGEN, NSET=HIGHTOP\n");
 fprintf(fp, " %d,%d, %d\n", Bn4,Bn5, Sinc);
 fprintf(fp, "*NFILL, NSET=NSURRA, TWO STEP,BIAS=%.3f\n", biasu);
 fprintf(fp, " SINGTOP,HIGHTOP, %d,%d\n", 2*Elabov,Binc);
 fprintf(fp, "*NSET, NSET=LEFTINN, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Sn6+Binc,Bn5, Binc);
 fprintf(fp, "*NSET, NSET=RITEINN, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Sn4+Binc,Bn4, Binc);
 fprintf(fp, "*NFILL, NSET=NSURRB, TWO STEP,BIAS=%.3f\n", biaso);
 if (Elbef != 0)
 {
  fprintf(fp, " SINGRITE,RITELOW, %d,%d\n", 2*Elbef,1);
 }
 fprintf(fp, " SINGLEFT,LEFTLOW, %d,%d\n", 2*Elaft,1);
 fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=LEFTINN,NEW SET=LEFTHIGH,
SHIFT,MULTIPLE=1\n", 2*Elaft);
 fprintf(fp, " -%.3f,%.3f,%.3f\n", width-(width*postn)-(singw/2),0,0);
 fprintf(fp, " 0.0,0.0,0.0, 0.0,0.0,1.0, 0.0\n");
 if (Elbef != 0)
 {
  fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=RITEINN,NEW SET=RITEHIGH,
SHIFT,MULTIPLE=1\n", 2*Elbef);
 fprintf(fp, " %.3f,%.3f,%.3f\n", width-(width*(1-postn))-(singw/2),0,0);
```

```
fprintf(fp, " 0.0,0.0,0.0, 0.0,0.0,1.0, 0.0\n");
}
fprintf(fp, "*NFILL, NSET=NSURRC, TWO STEP,BIAS=%.3f\n", biaso);
fprintf(fp, " LEFTINN,LEFTHIGH, %d,%d\n", 2*Elaft,1);
if (Elbef != 0)
{
 fprintf(fp, " RITEINN,RITEHIGH, %d,%d\n", 2*Elbef,1);
}
fprintf(fp, "**\n");
fprintf(fp, "**BOUNDARIES\n");
fprintf(fp, "*NSET, NSET=LOADEDGE, GENERATE\n");
if (Elbef != 0)
{
 fprintf(fp, " %d,%d, %d\n", Bn4,Bn3, 1);
}
fprintf(fp, " %d,%d, %d\n", Bn4,Bn5, Sinc);
fprintf(fp, " %d,%d, %d\n", Bn5,Bn6, 1);
fprintf(fp, "*NSET, NSET=CRFRONT, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", Sn9,Bn8, 1);
fprintf(fp, "**\n");

/* Calculate Element Quantities */
S  = ((2*Elsh)+Elsw)*con;
Ba = Elsh*Elbef;
Bb = Elbef*Elabov;
Bc = Elsw*Elabov;
Bd = Elabov*Elaft;
Be = Elsh*Elaft;

/* Continue Programme */
fprintf(fp, "**ELEMENT GENERATION\n");
fprintf(fp, "**SINGULARITY\n");
fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
fprintf(fp, " %d, %d,%d,%d,%d, %d,%d,%d,%d\n", 1, 1,3,3+(2*Sinc),1+(2*Sinc), 2,3+Sinc,2+(2*Sinc),1+Sinc);
fprintf(fp, "*ELGEN, ELSET=ELSING\n");
fprintf(fp, " %d, %d,%d,%d, %d,%d,%d\n", 1, con,2,1, (2*Elsh)+Elsw,2*Sinc,con);
fprintf(fp, "**BODY SURROUND\n");
fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
if (Elbef != 0.0)
{
 fprintf(fp, " %d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n",S+1, Sn2,Sn2+2,Sn2+2+(2*Sinc),Sn2+(2*Sinc),
Sn2+1,Sn2+2+Sinc,Sn2+1+(2*Sinc),Sn2+Sinc);
 fprintf(fp, " %d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n",1+S+Ba, Sn4,Sn4+2,Sn4+2+(2*Binc),Sn4+(2*Binc),
Sn4+1,Sn4+2+Binc,Sn4+1+(2*Binc),Sn4+Binc);
}
fprintf(fp, " %d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", S+Ba+Bb+1,
Sn4+(2*Sinc),Sn4,Sn4+(2*Binc),Sn4+(2*Sinc)+(2*Binc),
Sn4+Sinc,Sn4+Binc,Sn4+Sinc+(2*Binc),Sn4+(2*Sinc)+Binc);
fprintf(fp, " %d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", S+Ba+Bb+Bc+1,
Sn6+2,Sn6,Sn6+(2*Binc),Sn6+2+(2*Binc), Sn6+1,Sn6+Binc,Sn6+1+(2*Binc),Sn6+2+Binc);
fprintf(fp, " %d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", S+Ba+Bb+Bc+Bd+1,
Sn6+2+(2*Sinc),Sn6+(2*Sinc),Sn6,Sn6+2, Sn6+1+(2*Sinc),Sn6+Sinc,Sn6+1,Sn6+2+Sinc);
fprintf(fp, "*ELGEN, ELSET=ELSURR\n");
if (Elbef != 0.0)
{
 fprintf(fp, " %4d, %2d,%3d,%d, %2d,%4d,%2d\n", S+1, Elbef,2,1, Elsh,2*Sinc,Elbef);
 fprintf(fp, " %4d, %2d,%3d,%d, %2d,%4d,%2d\n", 1+S+Ba, Elbef,2,1, Elabov,2*Binc,Elbef);
}
fprintf(fp, " %4d, %2d,%3d,%d, %2d,%4d,%2d\n", 1+S+Ba+Bb,Elsw,2*Sinc,1, Elabov,2*Binc,Elsw);
fprintf(fp, " %4d, %2d,%3d,%d, %2d,%4d,%2d\n", 1+S+Ba+Bb+Bc,Elaft,2,1, Elabov,2*Binc,Elaft);
fprintf(fp, " %4d, %2d,%3d,%d, %2d,%4d,%2d\n", 1+S+Ba+Bb+Bc+Bd,Elaft,2,1, Elsh,2*Sinc,Elaft);
fprintf(fp, "*ELSET, ELSET=ALL\n");
fprintf(fp, " ELSING,ELSURR\n");
fprintf(fp, "**\n");
fprintf(fp, "**LOADINGS\n");
fprintf(fp, "*ELSET,ELSET=LOADED3, GENERATE\n");
if (Elbef != 0)
```

```
{
 fprintf(fp, " %d,%d, %d\n", S+Ba+Bb-Elbef+1,S+Ba+Bb, 1);
}
fprintf(fp, " %d,%d, %d\n", S+Ba+Bb+Bc-Elsw+1,S+Ba+Bb+Bc, 1);
fprintf(fp, " %d,%d, %d\n", S+Ba+Bb+Bc+Bd-Elaft+1,S+Ba+Bb+Bc+Bd, 1);
fprintf(fp, "**MATERIAL DEFINTION\n");
fprintf(fp, "*SOLID SECTION, ELSET=ALL,MATERIAL=AISI316\n");
fprintf(fp, " 4.0\n");
fprintf(fp, "*MATERIAL, NAME=AISI316\n");
fprintf(fp, "*DEFORMATION PLASTICITY\n");

/*Read from materials summary file - matsum */

if ((mat = fopen("matsum", "r")) == NULL)
{ printf("Cannot open materials summary file 'matsum'\n"); }

if (temp == 300) {line = 0;}
if (temp == 350) {line = 1;}
if (temp == 400) {line = 2;}
if (temp == 450) {line = 3;}
if (temp == 500) {line = 4;}
if (temp == 550) {line = 5;}
if (temp == 600) {line = 6;}
if (temp == 625) {line = 7;}
if (temp == 650) {line = 8;}
if (temp == 700) {line = 9;}

fseek(mat,(47*line)+197,0);
while ((inChar = getc(mat)) != '\n')
{ putc(inChar,fp); }

/* Continue Programme */
fprintf(fp, "\n**\n");
fprintf(fp, "**SINUSIODAL LOADING DEFINTION\n");
fprintf(fp, "*AMPLITUDE, DEFINITION=PERIODIC,VALUE=ABSOLUTE, NAME=SINWAVE\n");
fprintf(fp, " 1,6.283185307,0.0,-%.3f\n", Smean);
fprintf(fp, " 0.0,-%.1f, 0.0,0.0, 0.0,0.0, 0.0,0.0\n", Srange/2);
fprintf(fp, "**\n");
fprintf(fp, "*BOUNDARY\n");
fprintf(fp, " CRFRONT,  2\n");
fprintf(fp, " LOADEDGE, 1\n");
fprintf(fp, "**\n");
fprintf(fp, "**APPLY LINEAR RAMP TO MEAN STRESS\n");
fprintf(fp, "*STEP, INC=100\n");
fprintf(fp, "*STATIC\n");
fprintf(fp, " 0.1,1.0\n");
fprintf(fp, "*DLOAD, OP=MOD\n");
fprintf(fp, " LOADED3, P3, -%.3f\n", Smean);
fprintf(fp, "*CONTOUR INTEGRAL, CONTOUR=%d, SYMM, OUTPUT=BOTH\n", con);
fprintf(fp, " TIP, -1.0,0.0\n");
fprintf(fp, "*NODE PRINT, FREQUENCY=0\n");
fprintf(fp, "*EL PRINT, FREQUENCY=0\n");
fprintf(fp, "*NODE FILE, FREQUENCY=1\n");
fprintf(fp, " U\n");
fprintf(fp, "*EL FILE, FREQUENCY=1\n");
fprintf(fp, " S\n");
fprintf(fp, " E\n");
fprintf(fp, " PE\n");
fprintf(fp, "*END STEP\n");
fprintf(fp, "**\n");
fprintf(fp, "**APPLY SINUSIODAL LOADING PATTERN\n");
fprintf(fp, "*STEP, INC=100\n");
fprintf(fp, "*STATIC, DIRECT\n");
fprintf(fp, " 0.025,1.0\n");
fprintf(fp, "*DLOAD, AMPLITUDE=SINWAVE, OP=MOD\n");
fprintf(fp, " LOADED3, P3\n");
fprintf(fp, "*CONTOUR INTEGRAL, CONTOUR=%d, SYMM, OUTPUT=BOTH\n", con);
```

```
 fprintf(fp, " TIP, -1.0,0.0\n");
 fprintf(fp, "*NODE PRINT, FREQUENCY=0\n");
 fprintf(fp, "*EL PRINT, FREQUENCY=0\n");
 fprintf(fp, "*NODE FILE, FREQUENCY=5\n");
 fprintf(fp, " U\n");
 fprintf(fp, "*EL FILE, FREQUENCY=5\n");
 fprintf(fp, " S\n");
 fprintf(fp, " E\n");
 fprintf(fp, " PE\n");
 fprintf(fp, "*END STEP\n");

 crcnt++;
 postn = postn+crinc;
 fseek(read, 25+1+cnt, 1);
 fscanf(read, "%s", Elaftc);
 Elaft = atoi(Elaftc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elbefc);
 Elbef = atoi(Elbefc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elabovc);
 Elabov = atoi(Elabovc);
}


fclose(read);
fclose(fp);
fclose(mat);
return;
}
```

Appendix B.6

Programme Code: Cruciform Analysis Input File Generator

ABAQUS Input File Generator for Cruciform Fracture Model

```
/* Input File Generator - Multiple Files
  Cruciform Variable Crack Length */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{
 char  postnc[5], fname[20],fnamex[20], fadd[10];
 char  Elaftc[3],Elbefc[3],Elabovc[3], Elshc[3],Elswc[3], conc[2];
 char  biasoc[5],biasuc[5], singhc[4],singwc[4];
 char  ncracksc[2],crincc[5];
 int   inChar;
 int   Elaft,Elbef,Elabov, Elsh,Elsw, con;
 int   Sn1,Sn2,Sn3,Sn4,Sn5,Sn6,Sn7,Sn8,Sn9, Sinc;
 int   Bn1,Bn2,Bn3,Bn4,Bn5,Bn6,Bn7,Bn8, Bn9,Bn10,Bn11,Bn12, Binc, Tinc;
 int   OU1,OU2,OU3,OU4,OU5,OU6,OU7,OU8;
 int   Fhr1,Fhr2,Fhr3,Fhr4,Fhr5,Fhr6,Fhr7,Fhr8,Fhr9,Fhr10,Fhr11, NSOa,NSOb;
 int   Fvr1,Fvr2,Fvr3,Fvr4,Fvr5,Fvr6,Fvr7,Fvr8,Fvr9,Fvr10;
 int   flag;
 int   S, Ba,Bb,Bbb,Bbm,Bbt,Bc,Bcb,Bcm,Bct,Bd,Bdb,Bdm,Bdt,Bel,Bem,Ber,Be, temp,tcheck, line;
 int   ncracks, crcnt;
 int   Elnum1,Elnum2,Elnum3,Elnum4,Elnum5,Elnum6,Elnum7,Elnum8,Elnum9,Elnum10;
 int   Elnumh1,Elnumh2,Elnumh3,Elnumh4,Elnumh5,Elnumh6,Elnumh7;
 long  cnt;
 float postn, singh,singw, priormd;
 float biaso,biasu, Srange,Smean, fx1,fx2,fy2,fx7,fx8;
 float sx1,sy1, sx2,sy2, sx3,sy3, sx4,sy4, sx5,sy5,
     sx6,sy6, sx7,sy7, sx8,sy8, sx9,sy9,
     bx1,by1, bx2,by2, bx3,by3, bx4,by4, bx5,by5,
     bx6,by6, bx7,by7, bx8,by8, bx9,by9,
     oux1,ouy1, oux2,ouy2, oux3,ouy3, oux4,ouy4,
     oux5,ouy5, oux6,ouy6, oux7,ouy7, oux8,ouy8,
     fg2,fg3,fg4,fg5,fg6,fg7,fg8,fg9,fg10,fg11, NSOax,NSObx;
 float Gap,Fin;
 double Elgapv,Elfinv,Elhgapv, Elfinvl,Elgapvl,Elfinvlr;
 int    ElgapvI,ElfinvI,ElhgapvI, ElfinvlI,ElgapvlI,ElfinvlrI;
 int    ElfinhtI,ElgaphtI,ElfinhtbI,ElgaphbI,ElfinhbI,ElfinhbbI;
 double Elgapht,Elfinht,Elfinhtb, Elgaphb,Elfinhb,Elfinhbb,Elfinhbs;


 float crinc;
 float height, width;

 FILE *read;
 FILE *fp;
 FILE *mat;

 printf("\n\n*********************************************************\n");
 printf("Multiple Crack Length Input File Generator\n");
 printf("C Programme: progen.c\n");
 printf("Creates a file called 'input.inp' in current directory\n");
 printf("Uses mesh data from the file 'datasum' to run\n");
 printf("'datasum' file is generatoed by C Programme: meshgen.c\n");
 printf("*********************************************************\n\n");


 height = 25.0;
 width = 25.0;

/* Open data summary file for reading */
 if ((read = fopen("datasum", "r")) == NULL)
 { printf("Cannot open datasum\n"); }

/* Read & Write Fixed Element Data */
 cnt = 31;
```

```
fseek(read, 25+(1*cnt), 0);
fscanf(read, "%s", postnc);
postn = atof(postnc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", singhc);
singh = atof(singhc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", singwc);
singw = atof(singwc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", conc);
con = atoi(conc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", Elshc);
Elsh = atoi(Elshc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", Elswc);
Elsw = atof(Elswc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", biasoc);
biaso = atof(biasoc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", biasuc);
biasu = atof(biasuc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", ncracksc);
ncracks = atoi(ncracksc);

if (ncracks > 1)
{
 fseek(read, cnt+1, 1);
 fscanf(read, "%s", crincc);
 crinc = atof(crincc);
}

if (ncracks == 1)
{
 crinc = 0;
 fseek(read, 27+cnt, 1);
 fscanf(read, "%s", Elaftc);
 Elaft = atoi(Elaftc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elbefc);
 Elbef = atoi(Elbefc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elabovc);
 Elabov = atoi(Elabovc);
}

if (ncracks > 1)
{
 fseek(read, 26+1+cnt, 1);
 fscanf(read, "%s", Elaftc);
 Elaft = atoi(Elaftc);

 fseek(read, cnt+1, 1);
```

```
 fscanf(read, "%s", Elbefc);
 Elbef = atoi(Elbefc);

 fseek(read, cnt+1, 1);
 fscanf(read, "%s", Elabovc);
 Elabov = atoi(Elabovc);
 }


 printf("What is the Cyclic Stress Range (MPa, R=0.1): ");
 scanf(" %f", &Srange);
 Smean = (0.45*(Srange/0.9))+(0.1*(Srange/0.9));

 printf("What is the Temperature (celcius): ");
 scanf(" %d", &temp);
 printf("\n");

 tcheck = 0;
 while( tcheck == 0)
 {
  if (temp == 300)  { tcheck = 1; }
  if (temp == 350)  { tcheck = 1; }
  if (temp == 400)  { tcheck = 1; }
  if (temp == 450)  { tcheck = 1; }
  if (temp == 500)  { tcheck = 1; }
  if (temp == 550)  { tcheck = 1; }
  if (temp == 600)  { tcheck = 1; }
  if (temp == 625)  { tcheck = 1; }
  if (temp == 650)  { tcheck = 1; }
  if (temp == 700)  { tcheck = 1; }

  if (tcheck == 0)
  {
   printf("*** ERROR: Temperatures are only available for:\n");
   printf("         300, 350, 400, 450, 500, 550, 600, 625, 650, 700\n\n");
   printf("Please provide one the available Temperatures: ");
        scanf(" %d", &temp);
        printf("\n");
  }
 }


/* Start of Programme Loop For Multiple Crack Length Files */
 crcnt = 0;
 while (crcnt < ncracks)
 {
  if ( ncracks > 1)
  {
  printf("Provide a file name for crack %d at a/w = %.3f: ", crcnt+1,postn);
  scanf(" %s", fname);
  strcat(fname, ".inp");
  if ((fp = fopen(fname, "w")) == NULL)
  { printf("Cannot open %s\n", fname); }
  }
  if ( ncracks == 1)
  {
  if ((fp = fopen("input.inp", "w")) == NULL)
  { printf("Cannot open file input.inp\n"); }
  }

  /*Defme Element Quantities Thru Fingers and Gaps */
  /* Top Right Side */
  Elgapv = floor(7.2/(25.04/(Elaft+Elsw+Elbef)));
  Elfinv = floor((Elgapv/2.0)+0.5);
  Elhgapv = (Elaft+Elsw+Elbef)-(2*Elfmv)-(2*Elgapv);
  /*Top Left Side */
  Elgapvl = floor(7.2/(24.96/Elaft));
```

```
Elfinvl = floor((Elgapvl/2.0)+0.5);
Elfinvlr = Elaft-(2*Elgapvl)-Elfinvl;
/* Horizontal */
Elgapht = Elgapvl;
Elfinht = Elfinvl;
Elfinhtb = Elfinvlr;
Elgaphb = floor(7.2/(25.04/(Elabov+Elsh)));
Elfinhb = floor((Elgaphb/2.0)+0.5);
Elfinhbb = Elabov-((2*Elgaphb)+Elfinhb);
Elfinhbs = Elsh;

/* Define Singulrity Nodes */
Sinc = (2*con)+(2*(Elaft+Elaft+(int)Elfinvl+30+30))+1;
Sn1 = 1;
Sn2 = Sn1+(2*con);
Sn3 = Sn2+(Elsh*Sinc);
Sn4 = Sn3+(Elsh*Sinc);
Sn5 = Sn4+(Elsw*Sinc);
Sn6 = Sn5+(Elsw*Sinc);
Sn7 = Sn6+(Elsh*Sinc);
Sn8 = Sn7+(Elsh*Sinc);
Sn9 = Sn8-(2*con);

/* Define Body Nodes */
Bn1 = Sn2+(2*Elbef);
Bn2 = Bn1+(2*Elsh*Sinc);
Bn8 = Sn8+(2*Elaft);
Binc = Bn8+(2*(Elaft+(int)Elfinvl+30+30))-Sn4+1;

printf("Minimum Nodal Increment Upwards is %d\n", Binc);
printf("Provide an increment greater or equal to this: ");
scanf(" %d", &Tinc);
while (Tinc < Binc)
{
 printf("Increment must be greater than %d\n", Binc);
 printf("New Upwards Increment = ");
 scanf(" %d", &Tinc);
 printf("\n");
}
Binc = Tinc;

Bn3 = Bn2+(2*Elabov*Binc);
Bn4 = Bn3-(2*Elbef);
Bn5 = Bn4+(2*Elsw*Sinc);
Bn6 = Bn5+(2*Elaft);
Bn7 = Sn6+(2*Elaft);
Bn8 = Sn8+(2*Elaft);

/*Define Outer Square Uniform Nodes */
OU1 = Bn3+(2*Elaft*Binc);
OU2 = OU1-(2*Elbef);
OU3 = OU2+(2*Elsw*Sinc);
OU4 = OU3+(2*Elaft);
OU5 = OU4+(2*Elaft);
OU6 = Bn6+(2*Elaft);
OU7 = Bn7+(2*Elaft);
OU8 = Bn8+(2*Elaft);

priormd = (postn*width)-(singw/2);
printf("Prior Mesh Distance is %.3f\n\n", priormd);

fx1 = singh*0.25;
if (fx1 >= 0.40*priormd) { fx1 = 0.40*priormd; }
fx2 = singh*0.175;
if (fx2 >= 0.25*priormd) { fx2 = 0.25*priormd; }
fy2 = singw*0.15;
fx7 = singh*0.175;
```

```
fx8 = singh*0.25;


/* Singularity node co-ordinates */
sx1 = width*postn;
sy1 = 0.0;
sx2 = (width*postn)-(singw/2)-fx1;
if (Elbef == 0) { sx2 = 0.0; }
sy2 = 0.0;
sx3 = (width*postn)-(singw/2)-fx2;
if (Elbef == 0) { sx3 = 0.0; }
sy3 = singh/2;
sx4 = (width*postn)-(singw/2);
if (Elbef == 0) { sx4 = 0.0; }
sy4 = singh;
sx5 = (width*postn);
sy5 = singh+fy2;
sx6 = (width*postn)+(singw/2);
sy6 = singh;
sx7 = (width*postn)+(singw/2)+fx7;
sy7 = singh/2;
sx8 = (width*postn)+(singw/2)+fx8;
sy8 = 0.0;
sx9 = width*postn;
sy9 = 0.0;

/* Body node co-ordinates */
bx1 = 0.0;
by1 = 0.0;
bx2 = 0.0;
by2 = singh;
bx3 = 0.0;
by3 = height;
bx4 = (width*postn)-(singw/2);
by4 = height;
bx5 = (width*postn)+(singw/2);
by5 = height;
bx6 = width;
by6 = height;
bx7 = width;
by7 = singh;
bx8 = width;
by8 = 0.0;

/* Outer Square Uniform Co-ordinates */
oux1 =  0.000;
ouy1 = 46.480;
oux2 = ((25.0/(Elbef+Elsw+Elaft))*Elbef);
ouy2 = 46.480;
oux3 = ((25.0/(Elbef+Elsw+Elaft))*(Elbef+Elsw));
ouy3 = 46.480;
oux4 = 25.000;
ouy4 = 46.480;
oux5 = 46.480;
ouy5 = 46.480;
oux6 = 46.480;
ouy6 = 25.000;
oux7 = 46.480;
ouy7 =  3.600;
oux8 = 46.480;
ouy8 =  0.000;

/* Finger and Gap Positions */
Gap  = 7.200;
Fin  = 3.520;
fg2  = 3.600;
fg3  = fg2+Fin;
```

```
fg4  = fg3+Gap;
fg5  = fg4+Fin;
fg6  = fg5+Gap;
fg7  = fg6+Fin;
fg8  = fg7+Gap;
fg9  = fg8+Fin;
fg10 = fg9+Gap;
fg11 = 50.000;


/* Generate Abaqus Input File */

fprintf(fp, "*HEADING\n");
fprintf(fp, " ISOTHERMAL - %dC\n", temp);
fprintf(fp, " EQUIBIAXIAL - CYCLIC STRESS RANGE = %.1fMPa\n", Srange);
fprintf(fp, " %.3f A/W CRACK LENGTH\n", postn);
fprintf(fp, "**\n");
fprintf(fp, "*PREPRINT, ECHO=NO, MODEL=NO\n");
fprintf(fp, "**\n");
fprintf(fp, "**NODE GENERATION\n");
fprintf(fp, "**SINGULARITY\n");
fprintf(fp, "*NODE\n");
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn1, sx1,sy1);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn2, sx2,sy2);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn3, sx3,sy3);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn4, sx4,sy4);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn5, sx5,sy5);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn6, sx6,sy6);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn7, sx7,sy7);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn8, sx8,sy8);
fprintf(fp, "%5d, \t-%.3f, %.3f\n", Sn9, sx9,sy9);
fprintf(fp, "*NGEN, LINE=P, NSET=SINGRITE\n");
fprintf(fp, " %d,%d, %d,%d\n", Sn2,Sn4, Sinc,Sn3);
fprintf(fp, "*NGEN, LINE=P, NSET=SINGTOP\n");
fprintf(fp, " %d,%d, %d,%d\n", Sn4,Sn6, Sinc,Sn5);
fprintf(fp, "*NGEN, LINE=P, NSET=SINGLEFT\n");
fprintf(fp, " %d,%d, %d,%d\n", Sn6,Sn8, Sinc,Sn7);
fprintf(fp, "*NGEN, NSET=TIP\n");
fprintf(fp, " %d,%d, %d\n", Sn1,Sn9, Sinc);
fprintf(fp, "*NSET, NSET=SINGSURR\n");
fprintf(fp, " SINGRITE,SINGTOP,SINGLEFT\n");
fprintf(fp, "*NFILL, NSET=NSING,SINGULAR=1\n");
fprintf(fp, " TIP,SINGSURR, %d\n", 2*con,1);
fprintf(fp, "**\n");
fprintf(fp, "**BODY SURROUNDINGS\n");
fprintf(fp, "*NODE\n");
if (Elbef != 0)
{
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn1,bx1,by1);
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn2,bx2,by2);
 fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn3,bx3,by3);
}
fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn4,-bx4,by4);
fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn5,-bx5,by5);
fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn6,-bx6,by6);
fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn7,-bx7,by7);
fprintf(fp, "%7d, %8.3f,%8.3f\n", Bn8,-bx8,by8);
fprintf(fp, "*NGEN, NSET=RITELOW\n");
fprintf(fp, " %d,%d, %d\n", Bn1,Bn2, Sinc);
fprintf(fp, "*NGEN, NSET=LEFTLOW\n");
fprintf(fp, " %d,%d, %d\n", Bn7,Bn8, Sinc);
fprintf(fp, "*NGEN, NSET=HIGHTOP\n");
fprintf(fp, " %d,%d, %d\n", Bn4,Bn5, Sinc);
fprintf(fp, "*NFILL, NSET=NSURRA, TWO STEP,BIAS=%.3f\n", biasu);
fprintf(fp, " SINGTOP,HIGHTOP, %d,%d\n", 2*Elabov,Binc);
fprintf(fp, "*NSET, NSET=LEFTINN, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", Sn6+Binc,Bn5, Binc);
```

```
 fprintf(fp, "*NSET, NSET=RITEINN, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Sn4+Binc,Bn4, Binc);
 fprintf(fp, "*NFILL, NSET=NSURRB, TWO STEP,BIAS=%.3f\n", biaso);
 if (Elbef != 0)
 {
  fprintf(fp, " SINGRITE,RITELOW, %d,%d\n", 2*Elbef,1);
 }
 fprintf(fp, " SINGLEFT,LEFTLOW, %d,%d\n", 2*Elaft,1);
 fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=LEFTINN,NEW SET=LEFTHIGH,
SHIFT,MULTIPLE=1\n", 2*Elaft);
 fprintf(fp, " -%.3f,%.3f,%.3f\n", width-(width*postn)-(singw/2),0,0);
 fprintf(fp, " 0.0,0.0,0.0, 0.0,0.0,1.0, 0.0\n");
 if (Elbef != 0)
 {
  fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=RITEINN,NEW SET=RITEHIGH,
SHIFT,MULTIPLE=1\n", 2*Elbef);
  fprintf(fp, " %.3f,%.3f,%.3f\n", width-(width*(1-postn))-(singw/2),0,0);
  fprintf(fp, " 0.0,0.0,0.0, 0.0,0.0,1.0, 0.0\n");
 }
 fprintf(fp, "*NFILL, NSET=NSURRC, TWO STEP,BIAS=%.3f\n", biaso);
 fprintf(fp, " LEFTINN,LEFTHIGH, %d,%d\n", 2*Elaft,1);
 if (Elbef != 0)
 {
  fprintf(fp, " RITEINN,RITEHIGH, %d,%d\n", 2*Elbef,1);
 }
 fprintf(fp, "**\n");
 fprintf(fp, "**OUTER UNIFORM NODES FOR SQUARE\n");
 fprintf(fp, "**HORIZONTAL LINE\n");
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU1, -oux1,ouy1);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU2, -oux2,ouy2);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU3, -oux3,ouy3);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU4, -oux4,ouy4);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU5, -oux5,ouy5);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU6, -oux6,ouy6);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU7, -oux7,ouy7);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", OU8, -oux8,ouy8);
 fprintf(fp, "**TOP RIGHT CORNER\n");
 fprintf(fp, "*NGEN, NSET=OUHR\n");
 fprintf(fp, " %d,%d, %d\n", OU2,OU1, 1);
 fprintf(fp, " %d,%d, %d\n", OU2,OU3, Sinc);
 fprintf(fp, " %d,%d, %d\n", OU3,OU4, 1);
 fprintf(fp, "*NSET, NSET=MIDHR, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Bn4,Bn3, 1);
 fprintf(fp, " %d,%d, %d\n", Bn4,Bn5, Sinc);
 fprintf(fp, " %d,%d, %d\n", Bn5,Bn6, 1);
 fprintf(fp, "*NFILL, NSET=TOPR\n");
 fprintf(fp, " MIDHR,OUHR, %d,%d\n", 2*Elaft,Binc);
 fprintf(fp, "**BOTTOM LEFT\n");
 fprintf(fp, "*NGEN, NSET=OUVBB\n");
 fprintf(fp, " %d,%d, %d\n", OU7,OU8, Sinc);
 fprintf(fp, "*NGEN, NSET=OUVBT\n");
 fprintf(fp, " %d,%d, %d\n", OU7,OU6, Binc);
 fprintf(fp, "*NSET, NSET=OUVB\n");
 fprintf(fp, " OUVBB,OUVBT\n");
 fprintf(fp, "*NSET, NSET=VMIDB, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Bn7,Bn8, Sinc);
 fprintf(fp, " %d,%d, %d\n", Bn7,Bn6, Binc);
 fprintf(fp, "*NFILL, NSET=BOTL\n");
 fprintf(fp, " VMIDB,OUVB, %d,%d\n", 2*Elaft,1);
 fprintf(fp, "**TOP LEFT\n");
 fprintf(fp, "*NSET, NSET=HMIDL, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Bn6,OU6, 1);
 fprintf(fp, "*NGEN, NSET=OUHTOPL\n");
 fprintf(fp, " %d,%d, %d\n", OU4,OU5, 1);
 fprintf(fp, "*NFILL, NSET=TOPL\n");
 fprintf(fp, " HMIDL,OUHTOPL, %d,%d\n", 2*Elaft,Binc);
```

```
fprintf(fp, "**\n");
fprintf(fp, "**OUTER SQUARE EDGES SETUP FOR FINGERS\n");
fprintf(fp, "**HORIZONTAL LINE\n");

/* Position & Number of The Singularity Extension Nodes */
if ( ((Elbef+Elsw) == (Elhgapv+Elfinv+Elgapv))
  && (Elbef > (Elhgapv+Elfinv)) )
{ flag = 1; }
if ( ((Elbef+Elsw) < (Elhgapv+Elfinv+Elgapv))
  && (Elbef > (Elhgapv+Elfinv)) )
{ flag = 2; }
if ( ((Elbef+Elsw) < (Elhgapv+Elfinv+Elgapv))
  && (Elbef == (Elhgapv+Elfinv)) )
{ flag = 3; }
if ( ((Elbef+Elsw) > (Elhgapv+Elfinv))
  && (Elbef < (Elhgapv+Elfinv)) )
{ flag = 4; }
if ( ((Elbef+Elsw) == (Elhgapv+Elfinv))
  && (Elbef < (Elhgapv+Elfinv)) )
{ flag = 5; }
if ( ((Elbef+Elsw) < (Elhgapv+Elfinv))
  && (Elbef > Elhgapv) )
{ flag = 6; }
if ( ((Elbef+Elsw) < (Elhgapv+Elfinv))
  && (Elbef == Elhgapv) )
{ flag = 7; }
if ( ((Elbef+Elsw) > Elhgapv)
  && (Elbef < Elhgapv) )
{ flag = 8; }
if ( ((Elbef+Elsw) == Elhgapv)
  && (Elbef < Elhgapv) )
{ flag = 9; }
if ( ((Elbef+Elsw) < Elhgapv)
  && (Elbef < Elhgapv) )
{ flag = 10; }
if ( ((Elbef+Elsw) < Elhgapv)
  && (Elbef == 0) )
{ flag = 11; }

fprintf(fp, "**FLAG = %d\n", flag);


if (flag == 1)
{
 NSOax = 7.12+((7.2/Elgapv)*(Elbef-Elhgapv-Elfinv));
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = 14.32;
 NSOb  = (OU1+(Elfinht*2*Binc))-(2*Elbef)+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = Fhr1-(2*Elhgapv);
 Fhr3 = Fhr2-(2*Elfinv);
 Fhr4 = NSOb;
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr3, 1);
```

```
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr2, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 2)
{
 NSOax = 7.12+((7.2/Elgapv)*(Elbef-Elhgapv-Elfinv));
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = NSOax+((7.2/Elgapv)*Elsw);
 NSOb  = (OU1+(Elfinht*2*Binc))-(2*Elbef)+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = Fhr1-(2*Elhgapv);
 Fhr3 = Fhr2-(2*Elfinv);
 Fhr4 = NSOb+(2*((Elhgapv+Elfinv+Elgapv)-(Elbef+Elsw)));
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr3, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr2, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 3)
{
 NSOax = 7.12;
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = NSOax+((7.2/Elgapv)*Elsw);
 NSOb  = (OU1+(Elfinht*2*Binc))-(2*Elbef)+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = Fhr1-(2*Elhgapv);
 Fhr3 = NSOa;
 Fhr4 = NSOb+(2*((Elhgapv+Elfinv+Elgapv)-(Elbef+Elsw)));
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr2, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
```

```
}
if (flag == 4)
{
 NSOax = 3.6+((3.52/Elfinv)*(Elbef-Elhgapv));
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = 7.12+((7.2/Elgapv)*((Elbef+Elsw)-(Elhgapv+Elfinv)));
 NSOb  = (OU1+(Elfinht*2*Binc))-(2*Elbef)+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = Fhr1-(2*Elhgapv);
 Fhr3 = NSOa+(2*Sinc*((Elhgapv+Elfinv)-Elbef));
 Fhr4 = NSOb+(2*((Elhgapv+Elfinv+Elgapv)-(Elbef+Elsw)));
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr3, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr2, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 5)
{
 NSOax = 3.6+((3.52/Elfinv)*(Elbef-Elhgapv));
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = 7.12;
 NSOb  = NSOa+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = Fhr1-(2*Elhgapv);
 Fhr3 = NSOb;
 Fhr4 = NSOb+(2*Elgapv);
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr2, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 6)
{
 NSOax = 3.6+((3.52/Elfinv)*(Elbef-Elhgapv));
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
```

```
 NSObx = 3.6+((3.52/Elfinv)*((Elbef+Elsw)-Elhgapv));
 NSOb  = NSOa+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = Fhr1-(2*Elhgapv);
 Fhr3 = NSOb+(2*((Elhgapv+Elfinv)-(Elbef+Elsw)));
 Fhr4 = Fhr3+(2*Elgapv);
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr3, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr2, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}

if (flag == 7)
{
 NSOax = 3.6;
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = 3.6+((3.52/Elfinv)*((Elbef+Elsw)-Elhgapv));
 NSOb  = NSOa+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = NSOa;
 Fhr3 = NSOb+(2*((Elhgapv+Elfinv)-(Elbef+Elsw)));
 Fhr4 = Fhr3+(2*Elgapv);
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr3, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 8)
{
 NSOax = (3.6/Elhgapv)*Elbef;
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = 3.6+((3.52/Elfinv)*((Elbef+Elsw)-Elhgapv));
 NSOb  = NSOa+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
```

```
 Fhr2 = NSOa+(2*(Elhgapv-Elbef)*Sinc);
 Fhr3 = NSOb+(2*(Elaft-(Elgapv+Elfinv+Elgapv)));
 Fhr4 = Fhr3+(2*Elgapv);
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr1, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr2, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr3, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 9)
{
 NSOax = (3.6/Elhgapv)*Elbef;
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = 3.6;
 NSOb  = NSOa+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = NSOb;
 Fhr3 = NSOb+(2*Elfinv);
 Fhr4 = Fhr3+(2*Elgapv);
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr3, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 10)
{
 NSOax = (3.6/Elhgapv)*Elbef;
 NSOa  = (OU1+(Elfinht*2*Binc))-(2*Elbef);
 NSObx = (3.6/Elhgapv)*(Elbef+Elsw);
 NSOb  = NSOa+(2*Elsw*Sinc);

 Fhr1 = OU1+(2*Elfinht*Binc);
 Fhr2 = NSOb+(2*(Elhgapv-(Elbef+Elsw)));
 Fhr3 = Fhr2+(2*Elfinv);
 Fhr4 = Fhr3+(2*Elgapv);
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
```

```
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr1, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, -NSOax, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr2, 1);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOa,Fhr1, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr3, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}
if (flag == 11)
{
 NSOax = 0.0;
 NSOa  = (OU1+(Elfinht*2*Binc));
 NSObx = (3.6/Elhgapv)*Elsw;
 NSOb  = NSOa+(2*Elsw*Sinc);

 Fhr1 = NSOa;
 Fhr2 = NSOb+(2*(Elhgapv-Elsw));
 Fhr3 = Fhr2+(2*Elfinv);
 Fhr4 = Fhr3+(2*Elgapv);
 Fhr5 = Fhr4+(2*Elfinv);
 Fhr6 = Fhr5+(2*Elgapv);
 fprintf(fp, "*NODE\n");
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOa, 0.000,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", NSOb, -NSObx,50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr2, -fg2, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr3, -fg3, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr4, -fg4, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr5, -fg5, 50.000);
 fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr6, -fg6, 50.000);
 fprintf(fp, "*NGEN, NSET=OSQRBEF\n");
 fprintf(fp, " %7d,%7d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, " %7d,%7d, %d\n", NSOb,Fhr2, 1);
 fprintf(fp, "*NGEN, NSET=OSQRAFT\n");
 fprintf(fp, " %7d,%7d, %d\n", Fhr2,Fhr3, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr3,Fhr4, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr4,Fhr5, 1);
 fprintf(fp, " %7d,%7d, %d\n", Fhr5,Fhr6, 1);
}

fprintf(fp, "**LEFT HAND SIDE OF TOP OUTER EDGE\n");
Fhr7 = Fhr6+(2*Elfinvlr);
Fhr8 = Fhr7+(2*Elgapvl);
Fhr9 = Fhr8+(2*Elfinvl);
Fhr10 = Fhr9+(2*Elgapvl);
Fhr11 = Fhr10+(2*Elfinvl);
fg7 = fg6+Fin;
fg8 = fg7+Gap;
fg9 = fg8+Fin;
fg10 = fg9+Gap;
fg11 = fg10+Fin;
fprintf(fp, "*NODE\n");
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr7, -fg7, 50.000);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr8, -fg8, 50.000);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr9, -fg9, 50.000);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr10, -fg10, 50.000);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fhr11, -fg11, 50.000);
```

```
fprintf(fp, "*NGEN,NSET=OSQLL\n");
fprintf(fp, " %7d,%7d, %d\n", Fhr6,Fhr7, 1);
fprintf(fp, " %7d,%7d, %d\n", Fhr7,Fhr8, 1);
fprintf(fp, " %7d,%7d, %d\n", Fhr8,Fhr9, 1);
fprintf(fp, " %7d,%7d, %d\n", Fhr9,Fhr10, 1);

fprintf(fp, "*NSET, NSET=OUL, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", OU3,OU5, 1);
fprintf(fp, "*NSET, NSET=OSQL, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", NSOb,Fhr10, 1);
fprintf(fp, "*NFILL, NSET=HTOPSQ\n");
fprintf(fp, " OUL,OSQL, %.0lf,%d\n", 2*Elfinht,Binc);
fprintf(fp, "*NSET, NSET=OUM, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", OU2,OU3, Sinc);
fprintf(fp, "*NSET, NSET=OSQM, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", NSOa,NSOb, Sinc);
fprintf(fp, "*NFILL, NSET=OUSQM\n");
fprintf(fp, " OUM,OSQM, %.0lf,%d\n", 2*Elfinht,Binc);
fprintf(fp, "*NSET, NSET=OUR, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", OU2,OU1, 1);
fprintf(fp, "*NSET, NSET=OSQR, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", NSOa,Fhr1, 1);
fprintf(fp, "*NFILL, NSET=OUSQR\n");
fprintf(fp, " OUR,OSQR, %.0lf,%d\n", 2*Elfinht,Binc);


/* Nodal Generation for Left Vertical Edge */
Fvr1 = OU8+(2*Elfinvl);
Fvr2 = OU7+(2*Elfinvl);
Fvr3 = Fvr2+(2*Binc*Elfinhbb);
Fvr4 = Fvr3+(2*Binc*Elgaphb);
Fvr5 = Fvr4+(2*Binc*Elfinhb);
Fvr6 = Fvr5+(2*Binc*Elgaphb);
Fvr7 = Fvr6+(2*Binc*Elfinhtb);
Fvr8 = Fvr7+(2*Binc*Elgapht);
Fvr9 = Fvr8+(2*Binc*Elfinht);
Fvr10 = Fvr9+(2*Binc*Elgapht);
fprintf(fp, "**LEFT HAND SIDE VERTICAL OUTER EDGE\n");
fprintf(fp, "*NODE\n");
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr1, -50.000, 0.000);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr2, -50.000, fg2);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr3, -50.000, fg3);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr4, -50.000, fg4);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr5, -50.000, fg5);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr6, -50.000, fg6);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr7, -50.000, fg7);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr8, -50.000, fg8);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr9, -50.000, fg9);
fprintf(fp, " %7d, %7.3f,%7.3f\n", Fvr10, -50.000, fg10);
fprintf(fp, "*NGEN, NSET=OSQRV\n");
fprintf(fp, " %7d,%7d, %d\n", Fvr2,Fvr1, Sinc);
fprintf(fp, " %7d,%7d, %d\n", Fvr2,Fvr3, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr3,Fvr4, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr4,Fvr5, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr5,Fvr6, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr6,Fvr7, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr7,Fvr8, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr8,Fvr9, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr9,Fvr10, Binc);
fprintf(fp, " %7d,%7d, %d\n", Fvr10,Fhr11, Binc);
fprintf(fp, "*NSET, NSET=OUVBB, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", OU7,OU8, Sinc);
fprintf(fp, "*NSET, NSET=OSQVB, GENERATE\n");
fprintf(fp, " %d,%d, %d\n", Fvr2,Fvr1, Sinc);
fprintf(fp, "*NFILL, NSET=OUSQVB\n");
fprintf(fp, " OUVBB,OSQVB, %.0lf,%d\n", 2*Elfinvl,1);
fprintf(fp, "*NSET, NSET=OUVT, GENERATE\n");
```

```
 fprintf(fp, " %d,%d, %d\n", OU7,Fhr10, Binc);
 fprintf(fp, "*NSET, NSET=OSQVT, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Fvr2,Fhr11, Binc);
 fprintf(fp, "*NFILL, NSET=OUSQVT\n");
 fprintf(fp, " OUVT,OSQVT, %.0lf,%d\n", 2*Elfinvl,1);


 /* Generate Nodes For Vertical Fingers */
 fprintf(fp, "**\n");
 fprintf(fp, "**NODES FOR VERTICAL FINGERS\n");
 fprintf(fp, "*NSET, NSET=VFINBR, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", NSOa,Fhr1, 1);
 fprintf(fp, "*NSET, NSET=VFINBM, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", NSOa,NSOb, Sinc);
 fprintf(fp, "*NSET, NSET=VFINBL, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", NSOb,Fhr11, 1);
 fprintf(fp, "*NSET, NSET=VFINB\n");
 fprintf(fp, " VFINBR,VFINBM,VFINBL\n");
 fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=VFINB, SHIFT,MULTIPLE=1, NEW
SET=VFINM\n", 2*30*Binc);
 fprintf(fp, " %.3f,%.3f,%.3f\n", 0.0,25.0,0.0);
 fprintf(fp, " %.1f,%.1f,%.1f, %.1f,%.1f,%.1f, %.1f\n", 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0);
 fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=VFINM, SHIFT,MULTIPLE=1, NEW
SET=VFINT\n", 2*30*Binc);
 fprintf(fp, " %.3f,%.3f,%.3f\n", 0.0,55.0,0.0);
 fprintf(fp, " %.1f,%.1f,%.1f, %.1f,%.1f,%.1f, %.1f\n", 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0);
 fprintf(fp, "*NFILL, NSET=VFIN\n");
 fprintf(fp, " VFINB,VFINM, %d,%d\n", 2*30,Binc);
 fprintf(fp, " VFINM,VFINT, %d,%d\n", 2*30,Binc);
 /* Generate Nodes For Horizontal Fingers */
 fprintf(fp, "**\n");
 fprintf(fp, "**NODES FOR HORIZONTAL FINGERS\n");
 fprintf(fp, "*NSET, NSET=HFINRB, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Fvr2,Fvr1, Sinc);
 fprintf(fp, "*NSET, NSET=HFINRT, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Fvr2,Fhr11, Binc);
 fprintf(fp, "*NSET, NSET=HFINR\n");
 fprintf(fp, " HFINRB,HFINRT\n");
 fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=HFINR, SHIFT,MULTIPLE=1, NEW
SET=HFINM\n", 2*30);
 fprintf(fp, " %.3f,%.3f,%.3f\n", -25.0,0.0,0.0);
 fprintf(fp, " %.1f,%.1f,%.1f, %.1f,%.1f,%.1f, %.1f\n", 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0);
 fprintf(fp, "*NCOPY, CHANGE NUMBER=%d, OLD SET=HFINM, SHIFT,MULTIPLE=1, NEW
SET=HFINL\n", 2*30);
 fprintf(fp, " %.3f,%.3f,%.3f\n", -55.0,0.0,0.0);
 fprintf(fp, " %.1f,%.1f,%.1f, %.1f,%.1f,%.1f, %.1f\n", 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0);
 fprintf(fp, "*NFILL, NSET=HFIN\n");
 fprintf(fp, " HFINR,HFINM, %d,%d\n", 2*30,1);
 fprintf(fp, " HFINM,HFINL, %d,%d\n", 2*30,1);


 fprintf(fp, "**\n");
 fprintf(fp, "**BOUNDARIES\n");
 fprintf(fp, "*NSET, NSET=SYMMSIDE, GENERATE\n");
 if (priormd != 0)
 {
  fprintf(fp, " %d,%d, %d\n", Bn1,Bn2, Sinc);
  fprintf(fp, " %d,%d, %d\n", Bn2,Fhr1, Binc);
  fprintf(fp, " %d,%d, %d\n", Fhr1+(2*30*Binc),Fhr1+(2*30*Binc)+(2*30*Binc), Binc);
 }
 if (Elbef == 0)
 {
  fprintf(fp, " %d,%d, %d\n", Sn2,Sn4, Sinc);
  fprintf(fp, " %d,%d, %d\n", Sn4,Fhr1, Binc);
  fprintf(fp, " %d,%d, %d\n", Fhr1+(2*30*Binc),Fhr1+(2*30*Binc)+(2*30*Binc), Binc);
 }
 fprintf(fp, "*NSET, NSET=CRFRONT, GENERATE\n");
```

```
 fprintf(fp, " %d,%d, %d\n", Sn9,Fvr1, 1);
 fprintf(fp, " %d,%d, %d\n", Fvr1+(2*30),Fvr1+(2*(30+30)), 1);
 fprintf(fp, "*NSET, NSET=TIPSLAVE, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", Sn1+Sinc,Sn9, Sinc);
 fprintf(fp, "**\n");

 /* Calculate Element Quantities */
 S  = ((2*Elsh)+Elsw)*con;
 Ba  = Elsh*Elbef;
 Bbb = Elbef*Elabov;
 Bbm = Elbef*Elaft;
 Bbt = Elbef*(int)Elfinht;
 Bb  = Bbb+Bbm+Bbt;
 Bcb = Elsw*Elabov;
 Bcm = Elsw*Elaft;
 Bct = Elsw*(int)Elfinht;
 Bc  = Bcb+Bcm+Bct;
 Bdb = (Elaft+Elaft+(int)Elfinvl)*(Elabov+Elaft+(int)Elfinht);
 Bd  = Bdb+((Elabov+Elaft+Elfinht)*(30+30));
 Bel = Elsh*Elaft;
 Bem = Elsh*Elaft;
 Ber = Elsh*(int)Elfinvl;
 Be  = Bel+Bem+Ber+(Elsh*(30+30));


 /* Continue Programme */
 fprintf(fp, "**ELEMENT GENERATION\n");
 fprintf(fp, "**SINGULARITY\n");
 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %d, %d,%d,%d,%d, %d,%d,%d,%d,%d\n", 1, 1,3,3+(2*Sinc),1+(2*Sinc), 2,3+Sinc,2+(2*Sinc),1+Sinc);
 fprintf(fp, "*ELGEN, ELSET=ELSING\n");
 fprintf(fp, " %d, %d,%d,%d, %d,%d,%d\n", 1, con,2,1, (2*Elsh)+Elsw,2*Sinc,con);
 fprintf(fp, "**BODY SURROUND\n");
 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 if (Elbef != 0)
 {
  fprintf(fp, " %5d, %5d,%5d,%5d,%5d, %5d,%5d,%5d,%5d\n",S+1, Sn2,Sn2+2,Sn2+2+(2*Sinc),Sn2+(2*Sinc),
Sn2+1,Sn2+2+Sinc,Sn2+1+(2*Sinc),Sn2+Sinc);
  fprintf(fp, " %5d, %5d,%5d,%5d,%5d, %5d,%5d,%5d,%5d\n",1+S+Ba, Sn4,Sn4+2,Sn4+2+(2*Binc),Sn4+(2*Binc),
Sn4+1,Sn4+2+Binc,Sn4+1+(2*Binc),Sn4+Binc);
 }
 fprintf(fp, " %5d, %5d,%5d,%5d,%5d, %5d,%5d,%5d,%4d\n", S+Ba+Bb+1,
Sn4+(2*Sinc),Sn4,Sn4+(2*Binc),Sn4+(2*Sinc)+(2*Binc),
Sn4+Sinc,Sn4+Binc,Sn4+Sinc+(2*Binc),Sn4+(2*Sinc)+Binc);
 fprintf(fp, " %5d, %5d,%5d,%5d,%5d, %5d,%5d,%5d,%4d\n", S+Ba+Bb+Bc+1,
Sn6+2,Sn6,Sn6+(2*Binc),Sn6+2+(2*Binc), Sn6+1,Sn6+Binc,Sn6+1+(2*Binc),Sn6+2+Binc);
 fprintf(fp, " %5d, %5d,%5d,%5d,%5d, %5d,%5d,%5d,%4d\n", S+Ba+Bb+Bc+Bd+1,
Sn6+2+(2*Sinc),Sn6+(2*Sinc),Sn6,Sn6+2, Sn6+1+(2*Sinc),Sn6+Sinc,Sn6+1,Sn6+2+Sinc);
 fprintf(fp, "*ELGEN, ELSET=ELSURR\n");

 if (Elbef != 0)
 {
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", S+1, Elbef,2,1, Elsh,2*Sinc,Elbef);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", 1+S+Ba, Elbef,2,1, Elabov+Elaft+(int)Elfinht,2*Binc,Elbef);
 }
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", 1+S+Ba+Bb,Elsw,2*Sinc,1,
Elabov+Elaft+(int)Elfinht,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", 1+S+Ba+Bb+Bc,Elaft+Elaft+(int)Elfinvl,2,1,
Elabov+Elaft+(int)Elfinht,2*Binc,Elaft+Elaft+(int)Elfinvl+30+30);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", 1+S+Ba+Bb+Bc+Bd,Elaft+Elaft+(int)Elfinvl,2,1,
Elsh,2*Sinc,Elaft+Elaft+(int)Elfinvl+30+30);

 fprintf(fp, "**FINGER GENERATION\n");
 ElhgapvI  = (int)Elhgapv;
 ElfinvI   = (int)Elfinv;
 ElgapvI   = (int)Elgapv;
 ElfinvlrI = (int)Elfinvlr;
```

```
 ElgapvlI  = (int)Elgapvl;
 ElfinvlI  = (int)Elfinvl;

 if (flag == 1)
 {
  Elnum1 = S+Ba+Bb+Bc+Bd+Be+(Elbef-(ElhgapvI+ElfinvI))+1;
  Elnum2 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
  Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
  Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
  Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+1;
  Elnum6 = Elnum5+Elfinv+ElgapvI;
  Elnum7 = Elnum6+ElfinvlrI+ElgapvlI;
  Elnum8 = Elnum7+ElfinvlI+ElgapvlI;

  fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
Fhr3,Fhr3+2,Fhr3+2+(2*Binc),Fhr3+(2*Binc), Fhr3+1,Fhr3+2+Binc,Fhr3+1+(2*Binc),Fhr3+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
Fhr4+(2*30*Binc)+2,Fhr4+(2*30*Binc),Fhr4+(2*30*Binc)+(2*Binc),Fhr4+(2*30*Binc)+2,
Fhr4+(2*30*Binc)+1,Fhr4+(2*30*Binc)+Binc,Fhr4+(2*30*Binc)+1+(2*Binc),Fhr4+(2*30*Binc)+2+Binc);

  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

  fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, ElfinvI,2,1, 30,2*Binc,Elbef);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elbef,2,1, 30,2*Binc,Elbef);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }
 if (flag == 2)
 {
  Elnum1 = S+Ba+Bb+Bc+Bd+Be+(Elbef-(ElhgapvI+ElfinvI))+1;
  Elnum2 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
  Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
  Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
  Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
  Elnum6 = Elnum5+Elfinv+ElgapvI;
  Elnum7 = Elnum6+ElfinvlrI+ElgapvlI;
  Elnum8 = Elnum7+ElfinvlI+ElgapvlI;

  fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
Fhr3,Fhr3+2,Fhr3+2+(2*Binc),Fhr3+(2*Binc), Fhr3+1,Fhr3+2+Binc,Fhr3+1+(2*Binc),Fhr3+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
```

```
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2,
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, ElfinvI,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elbef,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }
 if (flag == 3)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
 Elnum6 = Elnum5+Elfinv+ElgapvI;
 Elnum7 = Elnum6+ElfinvlrI+ElgapvlI;
 Elnum8 = Elnum7+ElfinvlI+ElgapvlI;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa,NSOa+2,NSOa+2+(2*Binc),NSOa+(2*Binc), NSOa+1,NSOa+2+Binc,NSOa+1+(2*Binc),NSOa+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, ElfinvI,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elbef,2,1, 30,2*Binc,Elbef);
```

```
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }
 if (flag == 4)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
 Elnum6 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
 Elnum7 = Elnum6+Elfinv+ElgapvI;
 Elnum8 = Elnum7+ElfinvlrI+ElgapvlI;
 Elnum9 = Elnum8+ElfinvlI+ElgapvlI;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa,NSOa+2,NSOa+2+(2*Binc),NSOa+(2*Binc), NSOa+1,NSOa+2+Binc,NSOa+1+(2*Binc),NSOa+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*Sinc),NSOa,NSOa+(2*Binc),NSOa+(2*Sinc)+(2*Binc),
NSOa+Sinc,NSOa+Binc,NSOa+Sinc+(2*Binc),NSOa+(2*Sinc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum9,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elbef-ElhgapvI,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elbef,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, (ElhgapvI+ElfinvI)-Elbef,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum9, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }

 if (flag == 5)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
```

```
 Elnum6 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
 Elnum7 = Elnum6+Elfinv+ElgapvI;
 Elnum8 = Elnum7+ElfinvlrI+ElgapvlI;
 Elnum9 = Elnum8+ElfinvlI+ElgapvlI;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa,NSOa+2,NSOa+2+(2*Binc),NSOa+(2*Binc), NSOa+1,NSOa+2+Binc,NSOa+1+(2*Binc),NSOa+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*Sinc),NSOa,NSOa+(2*Binc),NSOa+(2*Sinc)+(2*Binc),
NSOa+Sinc,NSOa+Binc,NSOa+Sinc+(2*Binc),NSOa+(2*Sinc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum9,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elbef-ElhgapvI,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elbef,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, (ElhgapvI+ElfinvI)-Elbef,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum9, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }
 if (flag == 6)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+1;
 Elnum6 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
 Elnum7 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
 Elnum8 = Elnum7+Elfinv+ElgapvI;
 Elnum9 = Elnum8+ElfinvlrI+ElgapvlI;
 Elnum10 = Elnum9+ElfinvlI+ElgapvlI;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa,NSOa+2,NSOa+2+(2*Binc),NSOa+(2*Binc), NSOa+1,NSOa+2+Binc,NSOa+1+(2*Binc),NSOa+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*Sinc),NSOa,NSOa+(2*Binc),NSOa+(2*Sinc)+(2*Binc),
NSOa+Sinc,NSOa+Binc,NSOa+Sinc+(2*Binc),NSOa+(2*Sinc)+Binc);
```

```
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
NSOb+2,NSOb,NSOb+(2*Binc),NSOb+2+(2*Binc), NSOb+1,NSOb+Binc,NSOb+1+(2*Binc),NSOb+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum9,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum10,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elbef-Elhgapvl,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elbef,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, (Elhgapvl+Elfinvl)-(Elbef+Elsw),2,1,
30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, Elaft+Elaft+Elfinvll,2,1,
30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, Elfinvl,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, Elfinvlrl,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum9, Elfinvll,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum10, Elfinvll,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 }
 if (flag == 7)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+Elfinvll))+1;
 Elnum6 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(Elhgapvl+Elfinvl+Elgapvl-(Elbef+Elsw))+1;
 Elnum7 = Elnum6+Elfinv+Elgapvl;
 Elnum8 = Elnum7+Elfinvlrl+Elgapvll;
 Elnum9 = Elnum8+Elfinvll+Elgapvll;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*Sinc),NSOa,NSOa+(2*Binc),NSOa+(2*Sinc)+(2*Binc),
NSOa+Sinc,NSOa+Binc,NSOa+Sinc+(2*Binc),NSOa+(2*Sinc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOb+2,NSOb,NSOb+(2*Binc),NSOb+2+(2*Binc), NSOb+1,NSOb+Binc,NSOb+1+(2*Binc),NSOb+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
```

```
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum9,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elbef,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, (Elhgapvl+Elfinvl)-(Elbef+Elsw),2,1,
30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, Elaft+Elaft+Elfinvll,2,1,
30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, Elfinvl,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, Elfinvlrl,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, Elfinvll,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum9, Elfinvll,2,1, 30,2*Binc,Elaft+Elaft+Elfinvll);
 }
 if (flag == 8)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(Elhgapvl-Elbef)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+Elfinvll))+1;
 Elnum6 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(Elhgapvl+Elfinvl+Elgapvl-(Elbef+Elsw))+1;
 Elnum7 = Elnum6+Elfinv+Elgapvl;
 Elnum8 = Elnum7+Elfinvlrl+Elgapvll;
 Elnum9 = Elnum8+Elfinvll+Elgapvll;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
Fhr2+(2*Sinc),Fhr2,Fhr2+(2*Binc),Fhr2+(2*Sinc)+(2*Binc),
Fhr2+Sinc,Fhr2+Binc,Fhr2+Sinc+(2*Binc),Fhr2+(2*Sinc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOb+2,NSOb,NSOb+(2*Binc),NSOb+2+(2*Binc), NSOb+1,NSOb+Binc,NSOb+1+(2*Binc),NSOb+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum9,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elbef,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, (Elbef+Elsw)-Elhgapvl,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, (Elhgapvl+Elfinvl)-(Elbef+Elsw),2,1,
30,2*Binc,Elaft+Elaft+Elfinvll);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, Elaft+Elaft+Elfinvll,2,1,
30,2*Binc,Elaft+Elaft+Elfinvll);
```

```
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum9, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }
 if (flag == 9)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
 Elnum6 = Elnum5+Elfinv+ElgapvI;
 Elnum7 = Elnum6+ElfinvlrI+ElgapvlI;
 Elnum8 = Elnum7+ElfinvlI+ElgapvlI;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOb+2,NSOb,NSOb+(2*Binc),NSOb+2+(2*Binc), NSOb+1,NSOb+Binc,NSOb+1+(2*Binc),NSOb+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);

 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

 fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elbef,2,1, 30,2*Binc,Elbef);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }
 if (flag == 10)
 {
 Elnum1 = S+Ba+Bb+Bc+Bd+Be+(30*Elbef)+1;
 Elnum2 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
 Elnum3 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI-(Elbef+Elsw))+1;
 Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
 Elnum5 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
 Elnum6 = Elnum5+Elfinv+ElgapvI;
 Elnum7 = Elnum6+ElfinvlrI+ElgapvlI;
 Elnum8 = Elnum7+ElfinvlI+ElgapvlI;

 fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
 fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa+(2*30*Binc),NSOa+(2*30*Binc)+2,NSOa+(2*30*Binc)+2+(2*Binc),NSOa+(2*30*Binc)+(2*Binc),
NSOa+(2*30*Binc)+1,NSOa+(2*30*Binc)+2+Binc,NSOa+(2*30*Binc)+1+(2*Binc),NSOa+(2*30*Binc)+Binc);
```

```
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
Fhr2+2,Fhr2,Fhr2+(2*Binc),Fhr2+2+(2*Binc), Fhr2+1,Fhr2+Binc,Fhr2+1+(2*Binc),Fhr2+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);

  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum8,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

  fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elbef,2,1, 30,2*Binc,Elbef);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum8, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 }
 if (flag == 11)
 {
  Elnum1 = S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)+(30*Elsw)+1;
  Elnum2 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI-(Elbef+Elsw))+1;
  Elnum3 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(30*(Elaft+Elaft+ElfinvlI))+1;
  Elnum4 = S+Ba+Bb+Bc+Bd+Be+((Elbef+Elsw)*(30+30))+(ElhgapvI+ElfinvI+ElgapvI-(Elbef+Elsw))+1;
  Elnum5 = Elnum4+Elfinv+ElgapvI;
  Elnum6 = Elnum5+ElfinvlrI+ElgapvlI;
  Elnum7 = Elnum6+ElfinvlI+ElgapvlI;

  fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum1,
NSOa+(2*30*Binc)+(2*Sinc),NSOa+(2*30*Binc),NSOa+(2*30*Binc)+(2*Binc),NSOa+(2*30*Binc)+(2*Sinc)+(2*
Binc),
NSOa+(2*30*Binc)+Sinc,NSOa+(2*30*Binc)+Binc,NSOa+(2*30*Binc)+Sinc+(2*Binc),NSOa+(2*30*Binc)+(2*Si
nc)+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum2,
Fhr2+2,Fhr2,Fhr2+(2*Binc),Fhr2+2+(2*Binc), Fhr2+1,Fhr2+Binc,Fhr2+1+(2*Binc),Fhr2+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum3,
NSOb+(2*30*Binc)+2,NSOb+(2*30*Binc),NSOb+(2*30*Binc)+(2*Binc),NSOb+(2*30*Binc)+2+(2*Binc),
NSOb+(2*30*Binc)+1,NSOb+(2*30*Binc)+Binc,NSOb+(2*30*Binc)+1+(2*Binc),NSOb+(2*30*Binc)+2+Binc);

  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum4,
Fhr4+2,Fhr4,Fhr4+(2*Binc),Fhr4+(2*Binc)+2, Fhr4+1,Fhr4+Binc,Fhr4+1+(2*Binc),Fhr4+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum5,
Fhr6+2,Fhr6,Fhr6+(2*Binc),Fhr6+(2*Binc)+2, Fhr6+1,Fhr6+Binc,Fhr6+1+(2*Binc),Fhr6+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum6,
Fhr8+2,Fhr8,Fhr8+(2*Binc),Fhr8+(2*Binc)+2, Fhr8+1,Fhr8+Binc,Fhr8+1+(2*Binc),Fhr8+2+Binc);
  fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnum7,
Fhr10+2,Fhr10,Fhr10+(2*Binc),Fhr10+(2*Binc)+2, Fhr10+1,Fhr10+Binc,Fhr10+1+(2*Binc),Fhr10+2+Binc);

  fprintf(fp, "*ELGEN, ELSET=ELVERT\n");
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum1, Elsw,2*Sinc,1, 30,2*Binc,Elsw);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum2, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
  fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum3, Elaft+Elaft+ElfinvlI,2,1,
30,2*Binc,Elaft+Elaft+ElfinvlI);
```

```
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum4, ElfinvI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum5, ElfinvlrI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum6, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
 fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnum7, ElfinvlI,2,1, 30,2*Binc,Elaft+Elaft+ElfinvlI);
}

/* Horizontal Fingers */
fprintf(fp, "**\n");
fprintf(fp, "**HORIZONTAL FINGERS\n");

ElfinhtI  = (int)Elfinht;
ElgaphtI  = (int)Elgapht;
ElfinhtbI = (int)Elfinhtb;
ElgaphbI  = (int)Elgaphb;
ElfinhbI  = (int)Elfinhb;
ElfinhbbI  = (int)Elfinhbb;

Elnumh1 = S+Ba+Bb+Bc+Bd+Elaft+Elaft+ElfinvlI+30+1;
Elnumh2 = S+Ba+Bb+Bc+Elaft+Elaft+ElfinvlI+1;
Elnumh3 = Elnumh2+((Elaft+Elaft+ElfinvlI+30+30)*((ElfinhbbI+ElgaphbI)-1));
Elnumh4 = Elnumh3+((Elaft+Elaft+ElfinvlI+30+30)*((ElfinhbI+ElgaphbI)-1));
Elnumh5 = Elnumh4+((Elaft+Elaft+ElfinvlI+30+30)*((ElfinhtbI+ElgaphtI)-1));
Elnumh6 = Elnumh5+((Elaft+Elaft+ElfinvlI+30+30)*((ElfinhtI+ElgaphtI)-1));
Elnumh7 = Elnumh2+30;

fprintf(fp, "*ELEMENT, TYPE=CPS8R\n");
fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnumh1,
Fvr2+(2*30)+2+(2*Sinc),Fvr2+(2*30)+(2*Sinc),Fvr2+(2*30),Fvr2+(2*30)+2,
Fvr2+(2*30)+1+(2*Sinc),Fvr2+(2*30)+Sinc,Fvr2+(2*30)+1,Fvr2+(2*30)+2+Sinc);
fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnumh2,
Fvr2+2,Fvr2,Fvr2+(2*Binc),Fvr2+(2*Binc)+2, Fvr2+1,Fvr2+Binc,Fvr2+1+(2*Binc),Fvr2+2+Binc);
fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnumh3,
Fvr4+2,Fvr4,Fvr4+(2*Binc),Fvr4+(2*Binc)+2, Fvr4+1,Fvr4+Binc,Fvr4+1+(2*Binc),Fvr4+2+Binc);
fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnumh4,
Fvr6+2,Fvr6,Fvr6+(2*Binc),Fvr6+(2*Binc)+2, Fvr6+1,Fvr6+Binc,Fvr6+1+(2*Binc),Fvr6+2+Binc);
fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnumh5,
Fvr8+2,Fvr8,Fvr8+(2*Binc),Fvr8+(2*Binc)+2, Fvr8+1,Fvr8+Binc,Fvr8+1+(2*Binc),Fvr8+2+Binc);
fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnumh6,
Fvr10+2,Fvr10,Fvr10+(2*Binc),Fvr10+(2*Binc)+2, Fvr10+1,Fvr10+Binc,Fvr10+1+(2*Binc),Fvr10+2+Binc);
fprintf(fp, " %5d, %4d,%4d,%4d,%4d, %4d,%4d,%4d,%4d\n", Elnumh7,
Fvr2+(2*30)+2,Fvr2+(2*30),Fvr2+(2*30)+(2*Binc),Fvr2+(2*30)+2+(2*Binc),
Fvr2+(2*30)+1,Fvr2+(2*30)+Binc,Fvr2+(2*30)+1+(2*Binc),Fvr2+(2*30)+2+Binc);

fprintf(fp, "*ELGEN, ELSET=ELHORI\n");
fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnumh1, 30,2,1, Elsw,2*Sinc,(Elaft+Elaft+ElfinvlI+30+30));
fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnumh2, 30,2,1,
ElfinhbbI,2*Binc,(Elaft+Elaft+ElfinvlI+30+30));
fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnumh3, 30,2,1,
ElfinhbI,2*Binc,(Elaft+Elaft+ElfinvlI+30+30));
fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnumh4, 30,2,1,
ElfinhtbI,2*Binc,(Elaft+Elaft+ElfinvlI+30+30));
fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnumh5, 30,2,1,
ElfinhtI,2*Binc,(Elaft+Elaft+ElfinvlI+30+30));
fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnumh6, 30,2,1,
ElfinhtI,2*Binc,(Elaft+Elaft+ElfinvlI+30+30));
fprintf(fp, " %5d, %3d,%3d,%d, %3d,%5d,%3d\n", Elnumh7, 30,2,1,
Elabov+Elaft+ElfinhtI,2*Binc,(Elaft+Elaft+ElfinvlI+30+30));


fprintf(fp, "*ELSET, ELSET=ALL\n");
fprintf(fp, " ELSING,ELSURR,ELVERT,ELHORI\n");
fprintf(fp, "**\n");
fprintf(fp, "**LOADINGS\n");
fprintf(fp, "*ELSET,ELSET=LOADED3, GENERATE\n");
if (Elbef != 0)
{
```

```
 fprintf(fp, " %d,%d, %d\n", S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef)-
Elbef+1,S+Ba+Bb+Bc+Bd+Be+((30+30)*Elbef), 1);
 }
 fprintf(fp, " %d,%d, %d\n", S+Ba+Bb+Bc+Bd+Be+((30+30)*(Elbef+Elsw))-
Elsw+1,S+Ba+Bb+Bc+Bd+Be+((30+30)*(Elbef+Elsw)), 1);
 fprintf(fp, " %d,%d, %d\n", S+Ba+Bb+Bc+Bd+Be+((30+30)*(Elbef+Elsw+Elaft+Elaft+ElfinvlI))-
(Elaft+Elaft+ElfinvlI)+1,S+Ba+Bb+Bc+Bd+Be+((30+30)*(Elbef+Elsw+Elaft+Elaft+ElfinvlI)), 1);
 fprintf(fp, "*ELSET,ELSET=LOADED4, GENERATE\n");
 fprintf(fp, " %d,%d, %d\n", S+Ba+Bb+Bc+Bd+Elaft+Elaft+ElfinhtI+30+30,S+Ba+Bb+Bc+Bd+Be,
Elaft+Elaft+ElfinvlI+30+30);
 fprintf(fp, " %d,%d, %d\n", S+Ba+Bb+Bc+Elaft+Elaft+ElfinvlI+30+30,S+Ba+Bb+Bc+Bd,
Elaft+Elaft+ElfinvlI+30+30);
 fprintf(fp, "**\n");
 fprintf(fp, "**MATERIAL DEFINTION\n");
 fprintf(fp, "*SOLID SECTION, ELSET=ALL,MATERIAL=AISI316\n");
 fprintf(fp, " 4.0\n");
 fprintf(fp, "*MATERIAL, NAME=AISI316\n");
 fprintf(fp, "*DEFORMATION PLASTICITY\n");

 /*Read from materials summary file - matsum */

 if ((mat = fopen("matsum", "r")) == NULL)
 { printf("Cannot open materials summary file 'matsum'\n"); }

 if (temp == 300) {line = 0;}
 if (temp == 350) {line = 1;}
 if (temp == 400) {line = 2;}
 if (temp == 450) {line = 3;}
 if (temp == 500) {line = 4;}
 if (temp == 550) {line = 5;}
 if (temp == 600) {line = 6;}
 if (temp == 625) {line = 7;}
 if (temp == 650) {line = 8;}
 if (temp == 700) {line = 9;}

 fseek(mat,(47*line)+197,0);
 while ((inChar = getc(mat)) != '\n')
 { putc(inChar,fp); }

 /* Continue Programme */
 fprintf(fp, "\n**\n");
 fprintf(fp, "**SINUSIODAL LOADING DEFINTION\n");
 fprintf(fp, "*AMPLITUDE, DEFINITION=PERIODIC,VALUE=ABSOLUTE, NAME=SINWAVE\n");
 fprintf(fp, " 1,6.283185307,0.0,-%.3f\n", Smean);
 fprintf(fp, " 0.0,-%.1f, 0.0,0.0, 0.0,0.0, 0.0,0.0\n", Srange/2);
 fprintf(fp, "**\n");
 fprintf(fp, "*BOUNDARY\n");
 fprintf(fp, " SYMMSIDE, 1\n");
 fprintf(fp, " CRFRONT,  2\n");
 fprintf(fp, " 1,       2\n");
 fprintf(fp, " TIPSLAVE, 2\n");
 fprintf(fp, "*EQUATION\n");
 fprintf(fp, " 2\n");
 fprintf(fp, " TIPSLAVE,1,1.0, 1,1,-1.0\n");
 fprintf(fp, "**\n");
 fprintf(fp, "**APPLY LINEAR RAMP TO MEAN STRESS\n");
 fprintf(fp, "*STEP, INC=100\n");
 fprintf(fp, "*STATIC\n");
 fprintf(fp, " 0.1,1.0\n");
 fprintf(fp, "*DLOAD, OP=MOD\n");
 fprintf(fp, " LOADED3, P3, -%.3f\n", Smean);
 fprintf(fp, " LOADED4, P4, -%.3f\n", Smean);
 fprintf(fp, "*CONTOUR INTEGRAL, CONTOUR=%d, SYMM, OUTPUT=BOTH\n", con);
 fprintf(fp, " TIP, -1.0,0.0\n");
 fprintf(fp, "*NODE PRINT, FREQUENCY=0\n");
 fprintf(fp, "*EL PRINT, ELSET=LOADED3, FREQUENCY=6\n");
 fprintf(fp, " S22\n");
```

```
fprintf(fp, "*EL PRINT, ELSET=LOADED4, FREQUENCY=6\n");
fprintf(fp, " S11\n");
fprintf(fp, "*NODE FILE, FREQUENCY=0\n");
fprintf(fp, "*EL FILE, FREQUENCY=0\n");
fprintf(fp, "*RESTART, WRITE, OVERLAY\n");
fprintf(fp, "*END STEP\n");
fprintf(fp, "**\n");
fprintf(fp, "**APPLY SINUSIODAL LOADING PATTERN\n");
fprintf(fp, "*STEP, INC=100\n");
fprintf(fp, "*STATIC, DIRECT\n");
fprintf(fp, " 0.05,1.0\n");
fprintf(fp, "*DLOAD, AMPLITUDE=SINWAVE, OP=MOD\n");
fprintf(fp, " LOADED3, P3\n");
fprintf(fp, " LOADED4, P4\n");
fprintf(fp, "*CONTOUR INTEGRAL, CONTOUR=%d, SYMM, OUTPUT=BOTH\n", con);
fprintf(fp, " TIP, -1.0,0.0\n");
fprintf(fp, "*NODE PRINT, FREQUENCY=0\n");
fprintf(fp, "*EL PRINT, ELSET=LOADED3, FREQUENCY=5\n");
fprintf(fp, " S22\n");
fprintf(fp, "*EL PRINT, ELSET=LOADED4, FREQUENCY=5\n");
fprintf(fp, " S11\n");
fprintf(fp, "*NODE FILE, FREQUENCY=0\n");
fprintf(fp, "*EL FILE, FREQUENCY=0\n");
fprintf(fp, "*RESTART, WRITE, FREQUENCY=5\n");
fprintf(fp, "*END STEP\n");

crcnt++;
postn = postn+crinc;
fseek(read, 25+1+cnt, 1);
fscanf(read, "%s", Elaftc);
Elaft = atoi(Elaftc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", Elbefc);
Elbef = atoi(Elbefc);

fseek(read, cnt+1, 1);
fscanf(read, "%s", Elabovc);
Elabov = atoi(Elabovc);
}



fclose(read);
fclose(fp);
fclose(mat);
return;
}
```

Appendix B.7

ABAQUS Input File: Thermal Shock Analysis

ABAQUS Input File for Thermal Shock Fracture Analysis

(Stress Analysis)

```
*HEADING
 PURE THERMAL SHOCK - SINGLE EDGE
 BOUNDARY LOAD METHOD = 625.0C to 225.0C
 STRESS ANALYSIS
 0.025 A/W CRACK LENGTH
**
*RESTART, WRITE, FREQUENCY=1
*PREPRINT, ECHO=NO, MODEL=NO
**
**NODE GENERATION
**SINGULARITY
*NODE
    1,      -1.000, 0.000
   13,      -0.000, 0.000
   52,      -0.000, 0.500
   91,      -0.000, 1.000
  130,      -1.000, 1.150
  169,      -1.500, 1.000
  208,      -1.675, 0.500
  247,      -1.750, 0.000
  235,      -1.000, 0.000
*NGEN, LINE=P, NSET=SINGRITE
 13,91, 13,52
*NGEN, LINE=P, NSET=SINGTOP
 91,169, 13,130
*NGEN, LINE=P, NSET=SINGLEFT
 169,247, 13,208
*NGEN, NSET=TIP
 1,235, 13
*NSET, NSET=SINGSURR
 SINGRITE,SINGTOP,SINGLEFT
*NFILL, NSET=NSING,SINGULAR=1
 TIP,SINGSURR, 12
**
**BODY SURROUNDINGS
*NODE
     91,   -0.500,  30.000
    169,   -1.500,  30.000
    169,  -40.000,  30.000
    169,  -40.000,   1.000
    247,  -40.000,   0.000
*NGEN, NSET=LEFTLOW
 169,247, 13
*NGEN, NSET=HIGHTOP
 91,169, 13
*NFILL, NSET=NSURRA, TWO STEP,BIAS=0.960
 SINGTOP,HIGHTOP, 0,200
*NSET, NSET=LEFTINN, GENERATE
 369,169, 200
*NSET, NSET=RITEINN, GENERATE
 291,91, 200
*NFILL, NSET=NSURRB, TWO STEP,BIAS=0.944
 SINGLEFT,LEFTLOW, 0,1
*NCOPY, CHANGE NUMBER=0, OLD SET=LEFTINN,NEW SET=LEFTHIGH,
SHIFT,MULTIPLE=1
 -38.500,0.000,-0.000
 0.0,0.0,0.0, 0.0,0.0,1.0, 0.0
*NFILL, NSET=NSURRC, TWO STEP,BIAS=0.944
 LEFTINN,LEFTHIGH, 0,1
**
**BOUNDARIES
```

```
*NSET, NSET=LOADEDGE, GENERATE
 91,169, 13
 169,169, 1
*NSET, NSET=CRFRONT, GENERATE
 235,247, 1
**
**ELEMENT GENERATION
**SINGULARITY
*ELEMENT, TYPE=CPS8R
 1, 1,3,29,27, 2,16,28,14
*ELGEN, ELSET=ELSING
 1, 6,2,1, 9,26,6
**BODY SURROUND
*ELEMENT, TYPE=CPS8R
 55,  117,  91, 491, 517,  104, 291, 504, 317
 55,  171, 169, 569, 571,  170, 369, 570, 371
 55,  197, 195, 169, 171,  196, 182, 170, 184
*ELGEN, ELSET=ELSURR
   55,  3, 26,1,  0, 400, 3
   55,  0,  2,1,  0, 400, 0
   55,  0,  2,1,  3,  26, 0
*ELSET, ELSET=ALL
 ELSING,ELSURR
**
**LOADINGS
*ELSET,ELSET=LOADED3, GENERATE
 52,54, 1
 55,54, 1
**MATERIAL DEFINTION
*SOLID SECTION, ELSET=ALL,MATERIAL=AISI316
 4.0
*MATERIAL, NAME=AISI316
*DEFORMATION PLASTICITY
 189.748016E3,0.3,154.4,3.647467,0.174021, 300
 175.609222E3,0.3,146.9,3.423454,0.192785, 350
 171.887793E3,0.3,144.7,3.338149,0.186059, 400
 172.723328E3,0.3,147.2,3.547077,0.170800, 450
 180.182294E3,0.3,170.7,5.003758,0.043935, 500
 169.733842E3,0.3,161.9,4.776588,0.065672, 550
 166.792666E3,0.3,153.1,4.443184,0.098473, 600
 164.925896E3,0.3,150.3,4.515340,0.119600, 625
 167.212552E3,0.3,146.4,4.512722,0.111270, 650
 168.003465E3,0.3,131.4,4.639408,0.109740, 700
*EXPANSION, TYPE=ISO, ZERO=0
 16.2E-6, 300
 17.5E-6, 350
 17.5E-6, 400
 17.5E-6, 450
 17.5E-6, 500
 18.5E-6, 550
 18.5E-6, 600
 18.5E-6, 625
 18.5E-6, 650
 20.0E-6, 700
**
*BOUNDARY
 CRFRONT,  2
 LOADEDGE, 2
 91, 1
 169, 1
**
```

```
**APPLY LINEAR RAMP TO MEAN STRESS
*STEP, INC=100
*STATIC
*TEMPERATURE, FILE=input
*CONTOUR INTEGRAL, CONTOUR=6, SYMM, OUTPUT=BOTH
 TIP, -1.0,0.0
*NODE PRINT, FREQUENCY=0
*EL PRINT, FREQUENCY=0
*NODE FILE, FREQUENCY=1
 U
*EL FILE, FREQUENCY=1
 S
 E
 PE
*END STEP
```

Appendix B.8

ABAQUS Input File: Cruciform Analysis

ABAQUS Input File for Cruciform Fracture Analysis

```
*HEADING
 ISOTHERMAL - 625C
 EQUIBIAXIAL - CYCLIC STRESS RANGE = 200.0MPa
 0.025 A/W CRACK LENGTH
**
*PREPRINT, ECHO=NO, MODEL=NO
**
**NODE GENERATION
**SINGULARITY
*NODE
    1,       -0.625, 0.000
   13,       -0.000, 0.000
  412,       -0.000, 0.500
  811,       -0.000, 1.000
 1210,       -0.625, 1.150
 1609,       -1.125, 1.000
 2008,       -1.300, 0.500
 2407,       -1.375, 0.000
 2395,       -0.625, 0.000
*NGEN, LINE=P, NSET=SINGRITE
 13,811, 133,412
*NGEN, LINE=P, NSET=SINGTOP
 811,1609, 133,1210
*NGEN, LINE=P, NSET=SINGLEFT
 1609,2407, 133,2008
*NGEN, NSET=TIP
 1,2395, 133
*NSET, NSET=SINGSURR
 SINGRITE,SINGTOP,SINGLEFT
*NFILL, NSET=NSING,SINGULAR=1
 TIP,SINGSURR, 12
**
**BODY SURROUNDINGS
*NODE
    811,    -0.125,  25.000
   1609,    -1.125,  25.000
   1609,   -25.000,  25.000
   1609,   -25.000,   1.000
   2407,   -25.000,   0.000
*NGEN, NSET=RITELOW
 13,811, 133
*NGEN, NSET=LEFTLOW
 1609,2407, 133
*NGEN, NSET=HIGHTOP
 811,1609, 133
*NFILL, NSET=NSURRA, TWO STEP,BIAS=0.980
 SINGTOP,HIGHTOP, 0,2000
*NSET, NSET=LEFTINN, GENERATE
 3609,1609, 2000
*NSET, NSET=RITEINN, GENERATE
 2811,811, 2000
*NFILL, NSET=NSURRB, TWO STEP,BIAS=0.981
 SINGLEFT,LEFTLOW, 0,1
*NCOPY, CHANGE NUMBER=0, OLD SET=LEFTINN,NEW SET=LEFTHIGH,
SHIFT,MULTIPLE=1
 -23.875,0.000,-0.000
 0.0,0.0,0.0, 0.0,0.0,1.0, 0.0
*NFILL, NSET=NSURRC, TWO STEP,BIAS=0.981
 LEFTINN,LEFTHIGH, 0,1
**
**OUTER UNIFORM NODES FOR SQUARE
```

```
**HORIZONTAL LINE
*NODE
     811,    0.000, 46.480
     811,    0.000, 46.480
    1609, -25.000, 46.480
    1609, -25.000, 46.480
    1609, -46.480, 46.480
    1609, -46.480, 25.000
    1609, -46.480,  3.600
    2407, -46.480,  0.000
**TOP RIGHT CORNER
*NGEN, NSET=OUHR
 811,811, 1
 811,1609, 133
 1609,1609, 1
*NSET, NSET=MIDHR, GENERATE
 811,811, 1
 811,1609, 133
 1609,1609, 1
*NFILL, NSET=TOPR
 MIDHR,OUHR, 0,2000
**BOTTOM LEFT
*NGEN, NSET=OUVBB
 1609,2407, 133
*NGEN, NSET=OUVBT
 1609,1609, 2000
*NSET, NSET=OUVB
 OUVBB,OUVBT
*NSET, NSET=VMIDB, GENERATE
 1609,2407, 133
 1609,1609, 2000
*NFILL, NSET=BOTL
 VMIDB,OUVB, 0,1
**TOP LEFT
*NSET, NSET=HMIDL, GENERATE
 1609,1609, 1
*NGEN, NSET=OUHTOPL
 1609,1609, 1
*NFILL, NSET=TOPL
 HMIDL,OUHTOPL, 0,2000
**
**OUTER SQUARE EDGES SETUP FOR FINGERS
**HORIZONTAL LINE
**FLAG = 9
*NODE
     811,    0.000, 50.000
     811,    0.000, 50.000
    1609,   -3.600, 50.000
    1609,   -7.120, 50.000
    1609, -14.320, 50.000
    1609, -17.840, 50.000
    1609, -25.040, 50.000
*NGEN, NSET=OSQRBEF
    1609,    1609, 1
     811,    1609, 133
     811,     811, 1
*NGEN, NSET=OSQRAFT
    1609,    1609, 1
    1609,    1609, 1
    1609,    1609, 1
**LEFT HAND SIDE OF TOP OUTER EDGE
```

```
*NODE
    1609, -28.560, 50.000
    1609, -35.760, 50.000
    1609, -39.280, 50.000
    1609, -46.480, 50.000
    1609, -50.000, 50.000
*NGEN,NSET=OSQLL
    1609,   1609, 1
    1609,   1609, 1
    1609,   1609, 1
    1609,   1609, 1
*NSET, NSET=OUL, GENERATE
 1609,1609, 1
*NSET, NSET=OSQL, GENERATE
 1609,1609, 1
*NFILL, NSET=HTOPSQ
 OUL,OSQL, 0,2000
*NSET, NSET=OUM, GENERATE
 811,1609, 133
*NSET, NSET=OSQM, GENERATE
 811,1609, 133
*NFILL, NSET=OUSQM
 OUM,OSQM, 0,2000
*NSET, NSET=OUR, GENERATE
 811,811, 1
*NSET, NSET=OSQR, GENERATE
 811,811, 1
*NFILL, NSET=OUSQR
 OUR,OSQR, 0,2000
**LEFT HAND SIDE VERTICAL OUTER EDGE
*NODE
    2407, -50.000,  0.000
    1609, -50.000,  3.600
    1609, -50.000,  7.120
    1609, -50.000, 14.320
    1609, -50.000, 17.840
    1609, -50.000, 25.040
    1609, -50.000, 28.560
    1609, -50.000, 35.760
    1609, -50.000, 39.280
    1609, -50.000, 46.480
*NGEN, NSET=OSQRV
    1609,   2407, 133
    1609,   1609, 2000
    1609,   1609, 2000
    1609,   1609, 2000
    1609,   1609, 2000
    1609,   1609, 2000
    1609,   1609, 2000
    1609,   1609, 2000
    1609,   1609, 2000
    1609,   1609, 2000
*NSET, NSET=OUVBB, GENERATE
 1609,2407, 133
*NSET, NSET=OSQVB, GENERATE
 1609,2407, 133
*NFILL, NSET=OUSQVB
 OUVBB,OSQVB, 0,1
*NSET, NSET=OUVT, GENERATE
 1609,1609, 2000
*NSET, NSET=OSQVT, GENERATE
```

```
 1609,1609, 2000
*NFILL, NSET=OUSQVT
 OUVT,OSQVT, 0,1
**
**NODES FOR VERTICAL FINGERS
*NSET, NSET=VFINBR, GENERATE
 811,811, 1
*NSET, NSET=VFINBM, GENERATE
 811,1609, 133
*NSET, NSET=VFINBL, GENERATE
 1609,1609, 1
*NSET, NSET=VFINB
 VFINBR,VFINBM,VFINBL
*NCOPY, CHANGE NUMBER=120000, OLD SET=VFINB, SHIFT,MULTIPLE=1, NEW
SET=VFINM
 0.000,25.000,0.000
 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0
*NCOPY, CHANGE NUMBER=120000, OLD SET=VFINM, SHIFT,MULTIPLE=1, NEW
SET=VFINT
 0.000,55.000,0.000
 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0
*NFILL, NSET=VFIN
 VFINB,VFINM, 60,2000
 VFINM,VFINT, 60,2000
**
**NODES FOR HORIZONTAL FINGERS
*NSET, NSET=HFINRB, GENERATE
 1609,2407, 133
*NSET, NSET=HFINRT, GENERATE
 1609,1609, 2000
*NSET, NSET=HFINR
 HFINRB,HFINRT
*NCOPY, CHANGE NUMBER=60, OLD SET=HFINR, SHIFT,MULTIPLE=1, NEW
SET=HFINM
 -25.000,0.000,0.000
 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0
*NCOPY, CHANGE NUMBER=60, OLD SET=HFINM, SHIFT,MULTIPLE=1, NEW
SET=HFINL
 -55.000,0.000,0.000
 0.0,0.0,0.0, 0.0,1.0,0.0, 0.0
*NFILL, NSET=HFIN
 HFINR,HFINM, 60,1
 HFINM,HFINL, 60,1
**
**BOUNDARIES
*NSET, NSET=SYMMSIDE, GENERATE
 13,811, 133
 811,811, 2000
 120811,240811, 2000
 13,811, 133
 811,811, 2000
 120811,240811, 2000
*NSET, NSET=CRFRONT, GENERATE
 2395,2407, 1
 2467,2527, 1
*NSET, NSET=TIPSLAVE, GENERATE
 134,2395, 133
**
**ELEMENT GENERATION
**SINGULARITY
*ELEMENT, TYPE=CPS8R
```

```
 1, 1,3,269,267, 2,136,268,134
*ELGEN, ELSET=ELSING
 1, 6,2,1, 9,266,6
**BODY SURROUND
*ELEMENT, TYPE=CPS8R
    55,  1077,  811, 4811, 5077,   944, 2811, 4944,3077
    55,  1611, 1609, 5609, 5611,  1610, 3609, 5610,3611
    55,  1877, 1875, 1609, 1611,  1876, 1742, 1610,1744
*ELGEN, ELSET=ELSURR
    55,   3,266,1,   0, 4000,  3
    55,   0,  2,1,   0, 4000, 60
    55,   0,  2,1,   3,  266, 60
**FINGER GENERATION
*ELEMENT, TYPE=CPS8R
   235, 120811,120813,124813,124811, 120812,122813,124812,122811
   325, 121077,120811,124811,125077, 120944,122811,124944,123077
   415, 1611,1609,5609,5611, 1610,3609,5610,3611
   415, 121611,121609,125609,125611, 121610,123609,125610,123611
   415, 1611,1609,5609,5611, 1610,3609,5610,3611
   415, 1611,1609,5609,5611, 1610,3609,5610,3611
   415, 1611,1609,5609,5611, 1610,3609,5610,3611
   415, 1611,1609,5609,5611, 1610,3609,5610,3611
*ELGEN, ELSET=ELVERT
   235,   0,  2,1,  30, 4000,  0
   325,   3,266,1,  30, 4000,  3
   415,   0,  2,1,  30, 4000,  0
   415,   0,  2,1,  30, 4000,  0
   415,   0,  2,1,  30, 4000,  0
   415,   0,  2,1,  30, 4000,  0
   415,   0,  2,1,  30, 4000,  0
   415,   0,  2,1,  30, 4000,  0
**
**HORIZONTAL FINGERS
*ELEMENT, TYPE=CPS8R
    85, 1937,1935,1669,1671, 1936,1802,1670,1804
    55, 1611,1609,5609,5611, 1610,3609,5610,3611
    -5, 1611,1609,5609,5611, 1610,3609,5610,3611
   -65, 1611,1609,5609,5611, 1610,3609,5610,3611
  -125, 1611,1609,5609,5611, 1610,3609,5610,3611
  -185, 1611,1609,5609,5611, 1610,3609,5610,3611
    85, 1671,1669,5669,5671, 1670,3669,5670,3671
*ELGEN, ELSET=ELHORI
    85,  30,  2,1,   3,  266, 60
    55,  30,  2,1,   0, 4000, 60
    -5,  30,  2,1,   0, 4000, 60
   -65,  30,  2,1,   0, 4000, 60
  -125,  30,  2,1,   0, 4000, 60
  -185,  30,  2,1,   0, 4000, 60
    85,  30,  2,1,   0, 4000, 60
*ELSET, ELSET=ALL
 ELSING,ELSURR,ELVERT,ELHORI
**
**LOADINGS
*ELSET,ELSET=LOADED3, GENERATE
 412,414, 1
 415,414, 1
*ELSET,ELSET=LOADED4, GENERATE
 114,234, 60
 114,54, 60
**
**MATERIAL DEFINTION
```

```
*SOLID SECTION, ELSET=ALL,MATERIAL=AISI316
 4.0
*MATERIAL, NAME=AISI316
*DEFORMATION PLASTICITY
 164.925896E3,0.3,150.3,4.515340,0.119600, 625
**
**SINUSIODAL LOADING DEFINTION
*AMPLITUDE, DEFINITION=PERIODIC,VALUE=ABSOLUTE, NAME=SINWAVE
 1,6.283185307,0.0,-122.222
 0.0,-100.0, 0.0,0.0, 0.0,0.0, 0.0,0.0
**
*BOUNDARY
 SYMMSIDE, 1
 CRFRONT,  2
 1,        2
 TIPSLAVE, 2
*EQUATION
 2
 TIPSLAVE,1,1.0, 1,1,-1.0
**
**APPLY LINEAR RAMP TO MEAN STRESS
*STEP, INC=100
*STATIC
 0.1,1.0
*DLOAD, OP=MOD
 LOADED3, P3, -122.222
 LOADED4, P4, -122.222
*CONTOUR INTEGRAL, CONTOUR=6, SYMM, OUTPUT=BOTH
 TIP, -1.0,0.0
*NODE PRINT, FREQUENCY=0
*EL PRINT, ELSET=LOADED3, FREQUENCY=6
 S22
*EL PRINT, ELSET=LOADED4, FREQUENCY=6
 S11
*NODE FILE, FREQUENCY=0
*EL FILE, FREQUENCY=0
*RESTART, WRITE, OVERLAY
*END STEP
**
**APPLY SINUSIODAL LOADING PATTERN
*STEP, INC=100
*STATIC, DIRECT
 0.05,1.0
*DLOAD, AMPLITUDE=SINWAVE, OP=MOD
 LOADED3, P3
 LOADED4, P4
*CONTOUR INTEGRAL, CONTOUR=6, SYMM, OUTPUT=BOTH
 TIP, -1.0,0.0
*NODE PRINT, FREQUENCY=0
*EL PRINT, ELSET=LOADED3, FREQUENCY=5
 S22
*EL PRINT, ELSET=LOADED4, FREQUENCY=5
 S11
*NODE FILE, FREQUENCY=0
*EL FILE, FREQUENCY=0
*RESTART, WRITE, FREQUENCY=5
*END STEP
```

Appendix B.9

Programme Code: *UVARM* Fortran Code

Yield Zone Perimeter Stresses Written to ABAQUS .dat File for Extraction by Plastic Zone Extraction Programme

```
      SUBROUTINE UVARM(UVAR,DIRECT,T,TIME,DTIME,CMNAME,ORNAME,
     1 NUVARM,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,NDI,NSHR)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*8 CMNAME,ORNAME,FLGRAY(15)
      DIMENSION UVAR(NUVARM),DIRECT(3,3),T(3,3),TIME(2)
      DIMENSION ARRAY(15),JARRAY(15)
      REAL*8 PNTX(2,2), PNTY(2,2)
C
C Error Counter:
      JERROR = 0
C
C Get Stress Components:
      CALL GETVRM('S',ARRAY,JARRAY,FLGRAY,JRCD)
      JERROR = JERROR + JRCD
C
C Error message:
      IF(JERROR.NE.0)THEN
        WRITE(6,*) 'REQUESTED VARIABLE IS INAPPROPRIATE FOR ELNUM ',
     1    NOEL,'INTEGRATION POINT NUMBER ',NPT
      ENDIF
C
C Biaxiality Function:
C
      E    = 165D3
      ALPHA  = 18.5D-6
      DELTAT = 400
      MATCON = E*ALPHA*DELTAT
C
      Sx = ARRAY(1)
      Sy = ARRAY(2)
      YIELD = 200
C
      UVAR(1) = Sx*Sy/1500000
      UVAR(2) = Sx/Sy
      UVAR(3) = (2*Sx*Sy)/((Sx**2)+(Sy**2))
      UVAR(4) = ((2*Sx*Sy)/((Sx**2)+(Sy**2)))*Sy
      UVAR(5) = Sx-Sy
      UVAR(6) = Sy*(1-(Sx/sy))
C
C Test if Biaxial Quantity is Greater Than 1
      IF(UVAR(3).GE.1.0) THEN
        WRITE(6,*) 'S11 EQUALS ',ARRAY(1),' S22 EQUALS ',
     1    ARRAY(2),' & B EQUALS ',UVAR(3)
      ENDIF
C
C Test if Biaxial Quantity is Less Than 1
      IF(UVAR(3).LE.-1.0) THEN
        WRITE(6,*) 'S11 EQUALS ',ARRAY(1),' S22 EQUALS ',
     1    ARRAY(2),' & B EQUALS ',UVAR(3)
      ENDIF
C
C
C If at maximum increment or minimum increment
C Write integration point values to x and y arrays - PNTX & PNTY
      IF ( (KSTEP .EQ. 1) .OR. (NOEL .GT. 13000) ) RETURN
C
      IF ( (KINC .EQ. 25) .OR. (KINC .EQ. 75) ) THEN
        IF (NPT .EQ. 1) THEN
```

```
        PNTX(2,1) = Sx
        PNTY(2,1) = Sy
      ENDIF
      IF (NPT .EQ. 2) THEN
        PNTX(2,2) = Sx
        PNTY(2,2) = Sy
      ENDIF
      IF (NPT .EQ. 3) THEN
        PNTX(1,1) = Sx
        PNTY(1,1) = Sy
      ENDIF
      IF (NPT .EQ. 4) THEN
        PNTX(1,2) = Sx
        PNTY(1,2) = Sy
      ENDIF
C
      IF (NPT .EQ. 4) THEN
        xmin = PNTX(1,1)
        xmax = PNTX(1,1)
        ymin = PNTY(1,1)
        ymax = PNTY(1,1)
        IF ( PNTX(1,2) .LT. xmin ) THEN
          xmin = PNTX(1,2)
        ENDIF
        IF ( PNTX(2,1) .LT. xmin ) THEN
          xmin = PNTX(2,1)
        ENDIF
        IF ( PNTX(2,2) .LT. xmin ) THEN
         xmin = PNTX(2,2)
        ENDIF
        IF ( PNTX(1,2) .GT. xmax ) THEN
          xmax = PNTX(1,2)
        ENDIF
        IF ( PNTX(2,1) .GT. xmax ) THEN
          xmax = PNTX(2,1)
        ENDIF
        IF ( PNTX(2,2) .GT. xmax ) THEN
         xmax = PNTX(2,2)
        ENDIF
C
        IF ( PNTY(1,2) .LT. ymin ) THEN
          ymin = PNTY(1,2)
        ENDIF
        IF ( PNTY(2,1) .LT. ymin ) THEN
          ymin = PNTY(2,1)
        ENDIF
        IF ( PNTY(2,2) .LT. ymin ) THEN
         ymin = PNTY(2,2)
        ENDIF
        IF ( PNTY(1,2) .GT. ymax ) THEN
          ymax = PNTY(1,2)
        ENDIF
        IF ( PNTY(2,1) .GT. ymax ) THEN
          ymax = PNTY(2,1)
        ENDIF
        IF ( PNTY(2,2) .GT. ymax ) THEN
         ymax = PNTY(2,2)
        ENDIF
      ENDIF
C
```

```
C     Output element information to DAT file if above and below YIELD
C     After all integration points have been passed
      IF (NPT .EQ. 4) THEN
       IF ( (xmin .LT. YIELD) .AND. (xmax .GT. YIELD) ) THEN
         WRITE(6,*) 'X ELEMENT YIELDED  ELNUM ',NOEL
         WRITE(6,*) 'NTP 1 ',PNTX(2,1)
         WRITE(6,*) 'NTP 2 ',PNTX(2,2)
         WRITE(6,*) 'NTP 3 ',PNTX(1,1)
         WRITE(6,*) 'NTP 4 ',PNTX(1,2)
       ENDIF
       IF ( (ymin .LT. YIELD) .AND. (ymax .GT. YIELD) ) THEN
         WRITE(6,*) 'Y ELEMENT YIELDED  ELNUM ',NOEL
         WRITE(6,*) 'NTP 1 ',PNTY(2,1)
         WRITE(6,*) 'NTP 2 ',PNTY(2,2)
         WRITE(6,*) 'NTP 3 ',PNTY(1,1)
         WRITE(6,*) 'NTP 4 ',PNTY(1,2)
       ENDIF
      ENDIF
    ENDIF
C
C
    RETURN
    END
C
```

Appendix B.10

Programme Code: Plastic Zone Extraction

Data File Scanner to Extract and Tabulate Component Stress Plastic Zones from ABAQUS .dat File

```
/* Text Scanner For Nodal Locations
          using element number and integration point number
  Writes files by *zone flag with
     Element Number --- Yield Pt --- x coordinate --- y coordinate */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>


/* GLOBAL VARIABLES */
char elmndef[37]  = "E L E M E N T   D E F I N I T I O N S";
char ndef[31]     = "N O D E   D E F I N I T I O N S";
char inc6[15]     = "INCREMENT   6";
char yyield[27]   = "Y ELEMENT YIELDED  ELNUM  ";
char xyield[27]   = "X ELEMENT YIELDED  ELNUM  ";
char inc50[15]    = "INCREMENT   50";
char sign[2]      = " ";
char XorY[2]      = " ";


int  endlist;
long elrow = 1;

double pi = 3.14159265358979;


/* SUB-TROUTINES */
void timerstart();
void timerend();

void geteldata(char *yield, double sarray[500][5]);
void searcher(char *heading, int numch);
void elscan(double sarray[500][5], long elarray[500][5]);
void nscan(long elarray[500][5], double narray[2000][3]);
void intpnt(long elarray[500][5], double narray[500][3],
         double iarray[2000][4]);
void pzone(double iarray[2000][4], double sarray[500][5], char *file,
        char *when);


/* FILE OPERATORS */
FILE *data;
FILE *zone;


int main(int argc, char *argv[], char *env[])
{
 double xstress, ystress, inc;

 /* Maximum & Minimum x and y stresses */
 double sxmax[500][5],sxmin[500][5];
 double symax[500][5],symin[500][5];

 /* Maximum & Minimum x and y elements */
 long elxmax[500][5],elxmin[500][5];
 long elymax[500][5],elymin[500][5];
```

```
/* Maximum & Minimum x and y nodes    */
double nxmax[500][3],nxmin[500][3];
double nymax[500][3],nymin[500][3];

/* Maximum & Minimum x and y Integration Point Coordinates */
double ipxmax[4][4],ipxmin[4][4];
double ipymax[4][4],ipymin[4][4];

int i, doflag, dec,sign;
char *increment, *incstr, incmax[3] = "  ";

timerstart();

printf("\nARGC is %d\n",argc);
if (argc == 4)  { doflag = 1; }
if (argc == 6)  { doflag = 2; }
printf("\nDo flag is %d\n",doflag);
if ( (argc != 4) && (argc != 6) )
{
 printf("\n***OOOPS: Requires 3 or 5 arguments\n");
 printf("         file - inc - step - inc - step\n\n");
 exit(1);
}

if ((data = fopen(argv[1], "r")) == NULL)
{
 printf("***OOOPS: Cannot open file %s\n", argv[1]);
 printf("         It doesn't exist here.\n\n");
 printf("I have terminated myself\n\n");
 exit(1);
}

printf("These are the %d arguments passed to main\n\n", argc);
for (i = 0; i < argc; i++)
   printf("argv[%d]: %s\n", i, argv[i]);

/* GET MAXIMUM DATA - first inc/step set of given arguments */
if ( doflag <= 2)
{
 /* Make Increment search strings */
 inc = atoi(argv[2]);
 inc = (double)inc - 1;
 incstr = ecvt(inc,2,&dec,&sign);
 if ( dec == 1 )
 {
  incmax[1] = incstr[0];
  increment = strcat("INCREMENT     ",incmax);
  printf("\nSTRING IS %s\n",increment);
 }
 if ( dec == 2 )
 {
  increment = strcat("INCREMENT    ",incstr);
  printf("\nSTRING IS %s\n",increment);
 }

 /* Search out maximum loading x stresses and nodal points */
 fseek(data,0L,0);
 searcher(increment, 14);
 geteldata(xyield, sxmax);
 elscan(sxmax, elxmax);
```

```
 nscan(elxmax, nxmax);
 intpnt(elxmax, nxmax, ipxmax);
 pzone(ipxmax, sxmax, "xmax.txt", argv[2]);

 /* Search out maximum loading y stresses and nodal points */
 fseek(data,0L,0);
 searcher(increment, 14);
 geteldata(yyield, symax);
 elscan(symax, elymax);
 nscan(elymax, nymax);
 intpnt(elymax, nymax, ipymax);
 pzone(ipymax, symax, "ymax.txt", argv[2]);
 }

/* GET MINIMUM DATA */
 if ( doflag == 2 )
 {
 /* Make Increment search strings */
 inc = atoi(argv[3]);
 inc = (double)inc - 1;
 incstr = ecvt(inc,2,&dec,&sign);
 if ( dec == 1 )
 {
  incmax[1] = incstr[0];
  increment = strcat("INCREMENT     ",incmax);
  printf("\nSTRING IS %s\n",increment);
 }
 if ( dec == 2 )
 {
  increment = strcat("INCREMENT    ",incstr);
  printf("\nSTRING IS %s\n",increment);
 }
 /* Search out minimum loading x stresses and nodal points */
 fseek(data,0L,0);
 searcher(increment, 14);
 geteldata(xyield, sxmin);
 elscan(sxmin, elxmin);
 nscan(elxmin, nxmin);
 intpnt(elxmin, nxmin, ipxmin);
 pzone(ipxmin, sxmin, "xmin.txt", argv[3]);

 /* Search out minimum loading Y stresses and nodal points */
 fseek(data,0L,0);
 searcher(increment, 14);
 geteldata(yyield, symin);
 elscan(symin, elymin);
 nscan(elymin, nymin);
 intpnt(elymin, nymin, ipymin);
 pzone(ipymin, symin, "ymin.txt", argv[3]);
 }


 timerend();
 fclose(data);
 fclose(zone);
 printf("\nDONE\n");
 return;
}

/* SUB-ROUTINES */
```

```
/* Timers */
void timerstart()
{
 time_t t;

 time(&t);
 printf("Programme Began: %s\n", ctime(&t));
}
void timerend()
{
 time_t t;

 time(&t);
 printf("\n\nProgramme Completed: %s\n", ctime(&t));
}

/* String Searcher */
void searcher(char *heading, int numch)
{
 char start[37]  = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
 char spaces[11] = "xxxxxxxxxx";
 long n = 0;
 long m = 0;
 int  startflag = 0;

 while (startflag == 0)
 {
  start[n] = fgetc(data);
  if (start[n] == heading[n])
  {
   if (n == numch)  { startflag = 1; }
   n ++;
  }
  else
  {
   n = 0;
   if (start[0] == ' ')
   {
    if (m == 9) { endlist = 1; }
    m ++;
   }
   else
   {
    m = 0;
   }
  }

  if (start[n] == EOF)
  {
   startflag = 1;
   endlist = 1;
  }
 }

}

void geteldata(char *yield, double sarray[500][5])
{
 int check = 1;
 int cnt, flag;
```

```
 long element;

 double stress;

 printf("\n\nRetrieving Integration Point Stresses and Element Numbers\n");

 elrow = 1;
 while (check == 1)
 {
  endlist = 0;
  searcher(yield, 26);
  if ( (elrow > 1) && (endlist == 1) )  { break; }

  fscanf(data, "%ld", &element);
  sarray[elrow][1] = (double)element;

  cnt = 1;
  while (cnt <= 4 )
  {
   fseek(data, 11L,1);
   sign[0] = fgetc(data);
   if ( sign[0] == '-' ) { flag = 1; }
   else                  { flag = 0; }
   fscanf(data, "%lf", &stress);
   if ( flag == 1) { stress = -1*stress; }
   sarray[elrow][cnt+1] = stress;

   cnt++;
  }
  printf("\nElement: %7.0lf,  Stress: %8.3lf, %8.3lf, %8.3lf, %8.3lf",
   sarray[elrow][1],
   sarray[elrow][2],sarray[elrow][3],sarray[elrow][4],sarray[elrow][5]);

  fseek(data, 2L, 1);
  XorY[0] = fgetc(data);
  if ( (XorY[0] == 'Y') || (XorY[0] == 'X') )
  { check = 1; }
  if (XorY[0] == ' ')
  { break; }
  fseek(data, -3L, 1);

  if (elrow == 500)
  {
   printf("\n\n\n***** ELEMENT ROW ARRAY IS INADEQUATE AT 500 ROWS    *****");
   printf("\n\n\n***** PROGRAMME IS TERMINATED: INCREASE SYARRAY SIZE *****");
   printf("\n\n\n***** TO A VALUE UNRELEASED TO PEOPLE CALLED RICHARD\n\n\n *****");
   exit(1);
  }

  elrow++;
 }
 printf("\nNumber of elements found is %ld\n\n", elrow-1);
}

void elscan(double sarray[500][5], long elarray[500][5])
{
 char   *elstr1, elstr[17] = "xxxxxxxxxxxxxxxx";
 char   eladd[9] = "   CPS8R";
 char   nscanc[1] = "x";
 char   nstr[25];
```

```
int    len, dec,sign;
int    cnt, nflag,ncnt;
long   nnum;
long   elcnt = 1;

double  nnumd;

printf("\n\nRetrieving Elements and Defining Node Numbers\n");

while (elcnt < elrow)
{
 /* Construct search string */
 elstr1 = ecvt(sarray[elcnt][1],10,&dec,&sign);
 printf("\nELEMENT = %lf, STRING = %s, DEC = %d", sarray[elcnt][1],elstr1,dec);

 cnt = 0;
 while (cnt <= 9-dec-1)
 {
  elstr[cnt] = ' ';
  cnt ++;
 }
 cnt = 0;
 while (cnt <= dec-1)
 {
  elstr[9-dec+cnt] = elstr1[cnt];
  cnt++;
 }
 cnt = 0;
 while (cnt <= 7)
 {
  elstr[9+cnt] = eladd[cnt];
  cnt ++;
 }

 /* Search for string */
 fseek(data,0L,0);
 len = strlen(elstr);
 searcher(elstr,len-1);
 fseek(data,10L,1);

 /* Scan node numbers */
 ncnt = 1;
 while (ncnt <= 4)
 {
  nflag = 0;
  while (nflag == 0)
  {
   nscanc[0] = fgetc(data);
   if (nscanc[0] == ' ')  { nflag = 0; }
   else
   {
    nflag = 1;
    fseek(data, -1L,1);
   }
  }

  fscanf(data, "%ld", &nnum);
  elarray[elcnt][1] = (long)sarray[elcnt][1];
  elarray[elcnt][ncnt+1] = nnum;
```

```
 ncnt ++;
 }

 printf("\nElement %7ld, Nodes %8ld, %8ld, %8ld, %8ld   %4ld of %4ld",
  elarray[elcnt][1],
  elarray[elcnt][2],elarray[elcnt][3],elarray[elcnt][4],elarray[elcnt][5],
  elcnt,elrow-1);

 elcnt ++;
 }
}

void nscan(long elarray[500][5], double narray[2000][3])
{
 char *nstr1, nstr[9] = "       ";
 char cscan[1] = "x";

 int    len, dec,sign;
 int    cnt, ccnt,cflag;

 long   elcnt  = 1;
 long   ncnt   = 1;
 long   nnodes = 1;

 double  coord;


 printf("\n\n\nRetrieving Node Numbers and Defining x-y Coordinates\n");
 while (elcnt < elrow)
 {
  while (ncnt <= 4)
  {
   narray[nnodes][1] = (double)elarray[elcnt][1+ncnt];
   nstr1 = ecvt(narray[nnodes][1],10,&dec,&sign);

   /* Construct search string */
   cnt = 0;
   while (cnt <= 8-dec-1)
   {
    nstr[cnt] = ' ';
    cnt ++;
   }
   cnt = 0;
   while (cnt <= dec-1)
   {
    nstr[8-dec+cnt] = nstr1[cnt];
    cnt++;
   }

   /* Search for nodal co-ordinates starting from NODE DEFINITIONS*/
   fseek(data,0L,0);
   searcher(ndef,30);
   len = strlen(nstr);
   searcher(nstr,len-1);

   /* Scan co-ordinates */
   ccnt = 1;
   while (ccnt <= 2)
   {
```

```
   cflag = 0;
   while (cflag == 0)
   {
    cscan[0] = fgetc(data);
    if (cscan[0] == ' ')
    {
     cflag = 0;
    }
    else
    {
     if (cscan[0] == '-')
     {
      cflag = 2;
     }
     else
     {
      cflag = 1;
      fseek(data, -1L,1);
     }
    }
   }

   fscanf(data, "%lf", &coord);
   if (cflag == 1)  { narray[nnodes][ccnt+1] = coord; }
   if (cflag == 2)  { narray[nnodes][ccnt+1] = -1*coord; }

   ccnt ++;
   }

   printf("\nNode: %8.0lf,   XCoord: %9.5lf   YCoord: %9.5lf,   %4ld of %4ld",
    narray[nnodes][1],
    narray[nnodes][2],narray[nnodes][3],
    elcnt,elrow-1);

   ncnt ++;
   nnodes ++;
  }
  ncnt = 1;
  elcnt ++;
 }
}

void intpnt(long elarray[500][5], double narray[500][3], double iarray[4][4])
{
 long    search,ntest, elcnt,ncnt, nsearch, i,j,k,l;
 long    n1xf,n1yf, n2xf,n2yf, n3xf,n3yf, n4xf,n4yf;

 double  node, element, carray[4][2];

 double  n1x,n1y, n2x,n2y, n3x,n3y, n4x,n4y;
 double  L12,L23,L34,L41, hl12,hl23,hl34,hl41;
 double  D12,D23,D34,D41;
 double  A12,A23,A34,A41;
 double  IP112x,IP112y, IP212x,IP212y, IP223x,IP223y, IP423x,IP423y;
 double  IP434x,IP434y, IP334x,IP334y, IP341x,IP341y, IP141x,IP141y;
 double  IP1x,IP1y, IP2x,IP2y, IP3x,IP3y, IP4x,IP4y;
 double  m13,m24,m12,m34, C13,C24,C12,C34;


 printf("\n\nRetrieving Elements, Integration Point Stresses and Locations\n");
```

```
/* Find Nodal Co-ordinates */
elcnt = 1;
while (elcnt <= elrow-1)
{
 ncnt = 1;
 while (ncnt <= 4)
 {
 node = (double)elarray[elcnt][ncnt+1];
 /* Search for correct x y coords */
 nsearch = 0;
 search = 1;
 while (nsearch == 0)
 {
  ntest = narray[search][1];
  if (ntest == node)
  {
   carray[ncnt][1] = narray[search][2];
   carray[ncnt][2] = narray[search][3];
   nsearch = 1;
  }
  else { search ++; }
 }
 ncnt ++;
}

n1x = carray[1][1];  n1y = carray[1][2];
n2x = carray[2][1];  n2y = carray[2][2];
n3x = carray[3][1];  n3y = carray[3][2];
n4x = carray[4][1];  n4y = carray[4][2];

/* Determine Element Edge Lengths by Pythagoras */
L12 = sqrt(pow(n2x-n1x,2)+pow(n2y-n1y,2));
L23 = sqrt(pow(n3x-n2x,2)+pow(n3y-n2y,2));
L34 = sqrt(pow(n4x-n3x,2)+pow(n4y-n3y,2));
L41 = sqrt(pow(n1x-n4x,2)+pow(n1y-n4y,2));

/* Determine Half of Edge Lengths for Additions to
  Integration Point Distance                    */
hl12 = L12/2;
hl23 = L23/2;
hl34 = L34/2;
hl41 = L41/2;

/* Determine Integration Point Distance From Edge Centre */
D12 = L12/(2*sqrt(3));
D23 = L23/(2*sqrt(3));
D34 = L34/(2*sqrt(3));
D41 = L41/(2*sqrt(3));

/* Determine Angles of Edges in Relation to Horizontal and Vertical
  Global Axes                                          */
A12 = sqrt(pow(atan((n2y-n1y)/(n2x-n1x)),2));
A23 = sqrt(pow(atan((n3x-n2x)/(n3y-n2y)),2));
A34 = sqrt(pow(atan((n4y-n3y)/(n4x-n3x)),2));
A41 = sqrt(pow(atan((n4x-n1x)/(n4y-n1y)),2));

if (n1y == n2y) { A12 == 0.000; }
if (n2x == n3x) { A23 == 0.5*pi; }
if (n3y == n4y) { A34 == 0.000; }
```

```
if (n4x == n1x) { A41 == 0.5*pi; }

if (n1x == n2x) { A12 == 0.5*pi; }
if (n2y == n3y) { A23 == 0.000; }
if (n3x == n4x) { A34 == 0.5*pi; }
if (n4y == n1y) { A41 == 0.000; }

/* Determine +1 or -1 flags for perimeter point locations from
        Corner Nodes                                        */
/* X-FLAGS */
if (n2x > n1x) { n1xf =  1; } else { n1xf = -1; }
if (n3x > n2x) { n2xf =  1; } else { n2xf = -1; }
if (n4x < n3x) { n3xf = -1; } else { n3xf =  1; }
if (n4x > n1x) { n4xf = -1; } else { n4xf =  1; }

/* Y-FLAGS */
if (n2y > n1y) { n1yf =  1; } else { n1yf = -1; }
if (n3y > n2y) { n2yf =  1; } else { n2yf = -1; }
if (n4y > n3y) { n3yf =  1; } else { n3yf = -1; }
if (n4y > n1y) { n4yf = -1; } else { n4yf =  1; }

/* Determine Perimeter Points at Integration Point Distance
  From Edge Centre:  IP112 = IP1 on line 1 to 2          */
IP112x = n1x + (n1xf*(hl12-D12)*cos(A12));
IP112y = n1y + (n1yf*(hl12-D12)*sin(A12));
IP212x = n1x + (n1xf*(hl12+D12)*cos(A12));
IP212y = n1y + (n1yf*(hl12+D12)*sin(A12));

IP223x = n2x + (n2xf*(hl23-D23)*sin(A23));
IP223y = n2y + (n2yf*(hl23-D23)*cos(A23));
IP423x = n2x + (n2xf*(hl23+D23)*sin(A23));
IP423y = n2y + (n2yf*(hl23+D23)*cos(A23));

IP434x = n3x + (n3xf*(hl34-D34)*cos(A34));
IP434y = n3y + (n3yf*(hl34-D34)*sin(A34));
IP334x = n3x + (n3xf*(hl34+D34)*cos(A34));
IP334y = n3y + (n3yf*(hl34+D34)*sin(A34));

IP341x = n4x + (n4xf*(hl41-D41)*sin(A41));
IP341y = n4y + (n4yf*(hl41-D41)*cos(A41));
IP141x = n4x + (n4xf*(hl41+D41)*sin(A41));
IP141y = n4y + (n4yf*(hl41+D41)*cos(A41));

if ( L12 == 0 )
{ IP112x = n1x; IP112y = n1y; IP212x = n2x; IP212y = n2y; }
if ( L23 == 0 )
{ IP223x = n2x; IP223y = n2y; IP423x = n3x; IP423y = n3y; }
if ( L34 == 0 )
{ IP434x = n3x; IP434y = n3y; IP334x = n4x; IP334y = n4y; }
if ( L41 == 0 )
{ IP341x = n4x; IP341y = n4y; IP141x = n1x; IP141y = n1y; }

/* Determine Linear Equation Constants: Gradient 'm' & Intercapt 'C' */
m13 = (IP334y-IP112y)/(IP334x-IP112x);
m24 = (IP434y-IP212y)/(IP434x-IP212x);
m12 = (IP223y-IP141y)/(IP223x-IP141x);
m34 = (IP423y-IP341y)/(IP423x-IP341x);

C13 = IP112y - (m13*IP112x);
C24 = IP212y - (m24*IP212x);
```

```
 C12 = IP141y - (m12*IP141x);
 C34 = IP341y - (m34*IP341x);

 /* Determine x-y Co-ordinates of Integration Points by Bisection of
   Linear Equations                                          */
 IP1x = (C13-C12)/(m12-m13);   IP1y = (m12*IP1x)+C12;
 IP2x = (C24-C12)/(m12-m24);   IP2y = (m12*IP2x)+C12;
 IP3x = (C34-C13)/(m13-m34);   IP3y = (m13*IP3x)+C13;
 IP4x = (C34-C24)/(m24-m34);   IP4y = (m24*IP4x)+C24;

 /* If Divide by zero occurs from vertical gradient */
 if (IP334x == IP112x) { IP1x = IP112x; IP3x = IP334x; }
 if (IP434x == IP212x) { IP2x = IP212x; IP4x = IP434x; }
 if (IP141y == IP223y) { IP1y = IP141y; IP2y = IP223y; }
 if (IP341y == IP423y) { IP3y = IP341y; IP4y = IP423y; }


 /* Place Integration Point co-ordinates into iarray */
 i = (elcnt*4)-3; j = (elcnt*4) - 2; k = (elcnt*4)-1; l = (elcnt*4);
 element = (double)elarray[elcnt][1];

 iarray[i][1]=element; iarray[i][2]=1; iarray[i][3]=IP1x; iarray[i][4]=IP1y;
 iarray[j][1]=element; iarray[j][2]=2; iarray[j][3]=IP2x; iarray[j][4]=IP2y;
 iarray[k][1]=element; iarray[k][2]=3; iarray[k][3]=IP3x; iarray[k][4]=IP3y;
 iarray[l][1]=element; iarray[l][2]=4; iarray[l][3]=IP4x; iarray[l][4]=IP4y;

 printf("Element %lf, Int Pt. 1, x %lf, y %lf\n",
  iarray[i][1],iarray[i][3],iarray[i][4]);
 printf("Element %lf, Int Pt. 2, x %lf, y %lf\n",
  iarray[j][1],iarray[j][3],iarray[j][4]);
 printf("Element %lf, Int Pt. 3, x %lf, y %lf\n",
  iarray[k][1],iarray[k][3],iarray[k][4]);
 printf("Element %lf, Int Pt. 4, x %lf, y %lf\n",
  iarray[l][1],iarray[l][3],iarray[l][4]);

 elcnt ++;
 }
}

void pzone(double iarray[2000][4], double sarray[500][5], char *file,
        char *when)
{
 long    i,j,k,l;
 long    elcnt = 1;
 long    yptcnt = 1;

 double  yield = 200.0;
 double  IP1x,IP1y, IP2x,IP2y, IP3x,IP3y, IP4x,IP4y;
 double  s1,s2,s3,s4;
 double  factor, yptx,ypty;


 printf("\n\nWritting Elements, Yield Points and Coordintates to File %s\n\n",
  file);

 if ((zone = fopen(file, "w+")) == NULL)
 {
  printf("Cannot open file %s\n\n", file);
  exit(1);
 }
```

```
fprintf(zone, "  Element  Yield Pt        x        y  INCREMENT %s\n",
      when);

while (elcnt <= elrow-1)
{
/* Find Integration Point coords from iarray */
i = (elcnt*4)-3; j = (elcnt*4) - 2; k = (elcnt*4)-1; l = (elcnt*4);
IP1x = iarray[i][3];  IP1y = iarray[i][4];
IP2x = iarray[j][3];  IP2y = iarray[j][4];
IP3x = iarray[k][3];  IP3y = iarray[k][4];
IP4x = iarray[l][3];  IP4y = iarray[l][4];

/* Find Integration Point Stresses from sarray */
s1 = sarray[elcnt][2];
s2 = sarray[elcnt][3];
s3 = sarray[elcnt][4];
s4 = sarray[elcnt][5];

/* Determine Yield Line From Stresses at Integration Points */
if ( ((s1 < yield) && (s2 > yield)) || ((s2 < yield) && (s1 > yield)) )
{
 factor = (yield-s1)/(s2-s1);
 yptx = IP1x + (factor*(IP2x-IP1x));
 ypty = IP1y + (factor*(IP2y-IP1y));
 if ( (yptx > -25.0 ) && (ypty < 25.0) )
 {
  fprintf(zone, "%10.0lf %9ld %10lf %10lf\n",
   sarray[elcnt][1],yptcnt,yptx,ypty);
  printf("Element %lf IP1 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP1x,IP1y, s1);
  printf("Element %lf IP2 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP2x,IP2y, s2);
  printf("Element %lf IP3 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP3x,IP3y, s3);
  printf("Element %lf IP4 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP4x,IP4y, s4);
  printf("Element %10.0lf Yield Pt. %10ld x %11lf y %11lf\n",
   sarray[elcnt][1],yptcnt,yptx,ypty);
 }
 yptcnt ++;
}
if ( ((s2 < yield) && (s4 > yield)) || ((s4 < yield) && (s2 > yield)) )
{
 factor = (yield-s2)/(s4-s2);
 yptx = IP2x + (factor*(IP4x-IP2x));
 ypty = IP2y + (factor*(IP4y-IP2y));
 if ( (yptx > -25.0 ) && (ypty < 25.0) )
 {
  fprintf(zone, "%10.0lf %9ld %10lf %10lf\n",
   sarray[elcnt][1],yptcnt,yptx,ypty);
  printf("Element %lf IP1 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP1x,IP1y, s1);
  printf("Element %lf IP2 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP2x,IP2y, s2);
  printf("Element %lf IP3 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP3x,IP3y, s3);
  printf("Element %lf IP4 x %lf y %lf stress %lf\n",
   sarray[elcnt][1], IP4x,IP4y, s4);
  printf("Element %10.0lf Yield Pt. %10ld x %11lf y %11lf\n",
```

```
   sarray[elcnt][1],yptcnt,yptx,ypty);
  }
  yptcnt ++;
 }
 if ( ((s4 < yield) && (s3 > yield)) || ((s3 < yield) && (s4 > yield)) )
 {
  factor = (yield-s4)/(s3-s4);
  yptx = IP4x + (factor*(IP3x-IP4x));
  ypty = IP4y + (factor*(IP3y-IP4y));
  if ( (yptx > -25.0 ) && (ypty < 25.0) )
  {
   fprintf(zone, "%10.0lf %9ld %10lf %10lf\n",
    sarray[elcnt][1],yptcnt,yptx,ypty);
   printf("Element %lf IP1 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP1x,IP1y, s1);
   printf("Element %lf IP2 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP2x,IP2y, s2);
   printf("Element %lf IP3 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP3x,IP3y, s3);
   printf("Element %lf IP4 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP4x,IP4y, s4);
   printf("Element %10.0lf Yield Pt. %10ld x %11lf y %11lf\n",
    sarray[elcnt][1],yptcnt,yptx,ypty);
  }
  yptcnt ++;
 }
 if ( ((s3 < yield) && (s1 > yield)) || ((s1 < yield) && (s3 > yield)) )
 {
  factor = (yield-s3)/(s1-s3);
  yptx = IP3x + (factor*(IP1x-IP3x));
  ypty = IP3y + (factor*(IP1y-IP3y));
  if ( (yptx > -25.0 ) && (ypty < 25.0) )
  {
   fprintf(zone, "%10.0lf %9ld %10lf %10lf\n",
    sarray[elcnt][1],yptcnt,yptx,ypty);
   printf("Element %lf IP1 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP1x,IP1y, s1);
   printf("Element %lf IP2 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP2x,IP2y, s2);
   printf("Element %lf IP3 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP3x,IP3y, s3);
   printf("Element %lf IP4 x %lf y %lf stress %lf\n",
    sarray[elcnt][1], IP4x,IP4y, s4);
   printf("Element %10.0lf Yield Pt. %10ld x %11lf y %11lf\n",
    sarray[elcnt][1],yptcnt,yptx,ypty);
  }
  yptcnt ++;
 }

 yptcnt = 1;
 elcnt ++;
}

}
```

Appendix B.11

Programme Code: ASTM E647

Matlab Polynomial Solver for Determination of da/dN Crack Growth Rates from a-N Data

```
function [a,N,ahat,cgr] = crucgr(name,range)
% Fits Measured Crack Lenghts to a 3rd Order Polynomial
%                   against N (Number of Cycles) by 7pt cycling.
% Data set loaded from a text file with N data
%                   in the 1st column - type name without .ext
% Then places the fitted crack lengths into ahat array
%                   and plots them together.
% Finally Calculates Crack Growth Rates and Plots them
%                   with output to a file cgr.txt

% Set raw data
fname = strcat(name,'.txt');
data = load (fname);

a = data(:,2);
N = data(:,1);
rows = size(N,1);
rows = rows-6;


% Fit 3rd Order Polynomial as for 7pt cycling
for i = 1:rows
  afit = polyfit(N(i:i+6,1),a(i:i+6,1),2);
  ahat(i+3,1) = polyval(afit,N(i+3,1));
  trig = cos(pi*ahat(i+3,1)*0.001/0.1);
  dK(i+3,1) = (range*1e6*(sqrt(pi*ahat(i+3,1)*0.001/trig)))/1e6;
  rate(i+3,1) = afit(1,2)+(2*afit(1,1)*N(i+3,1));
end

% Add Trialing Zeros to End of ahat & rate array to maintain equal size
for i = 1:3
   ahat(rows+3+i,1) = 0;
   dK(rows+3+i,1) = 0;
            rate(rows+3+i,1) = 0;
end

% Assign Values to a Crack Growth Rate Array - cgr
cgr(:,:) = [ahat,dK,rate];
save cgr.txt cgr -ascii;

figure(1);
plot(N(4:rows+3,1),a(4:rows+3,1),'bd',N(4:rows+3,1),ahat(4:rows+3,1),'r-');
figure(2);
loglog(cgr(4:rows-3,2),cgr(4:rows-3,3),'bd');
```